Stress-Plus-X (SPX) Graph Layout

Sabin Devkota(✉)[0000−0002−0610−6573], Reyan Ahmed[0000−0001−6830−9053], Felice De Luca[0000−0001−5937−7636], Katherine E. Isaacs[0000−0002−9947−928X], Stephen Kobourov[0000−0002−0477−2724]

Department of Computer Science, University of Arizona, USA
{devkotasabin, abureyanahmed, felicedeluca}@email.arizona.edu,
{kisaacs, kobourov}@cs.arizona.edu

Abstract. Stress, edge crossings, and crossing angles play an important role in the quality and readability of graph drawings. Most standard graph drawing algorithms optimize one of these criteria which may lead to layouts that are deficient in other criteria. We introduce an optimization framework, Stress-Plus-X (SPX), that simultaneously optimizes stress together with several other criteria: edge crossings, minimum crossing angle, and upwardness (for directed acyclic graphs). SPX achieves results that are close to the state-of-the-art algorithms that optimize these metrics individually. SPX is flexible and extensible and can optimize a subset or all of these criteria simultaneously. Our experimental analysis shows that our joint optimization approach is successful in drawing graphs with good performance across readability criteria.

1 Introduction

Several criteria have been proposed for evaluating the quality of graph layouts [37], including minimizing stress, minimizing the number of edge crossings, minimizing drawing area, as well as maximizing the angle between edge crossings, maintaining separation between marks ("resolution"), and preserving highly connected neighborhoods. In the case of directed acyclic graphs (DAGs), maintaining consistent edge direction, i.e., upwardness, is preferable. While these criteria have been shown to improve human performance for graph tasks, automatic layout approaches actively target at most one from the list.

We propose a framework, *Stress-Plus-X* (*SPX*), for automatic layout of node-link diagrams that targets multiple graph layout criteria simultaneously. SPX formulates the layout as an optimization problem that combines stress minimization with penalty terms representing other criteria. Composing and weighting the terms in the objective function provides the flexibility and extensibility.

With the adage *"Don't let perfect be the enemy of good"* in mind, the goal of SPX is not to optimize any one particular criterion at the cost of all others, but to find a balance across the criteria as optimizing only one criterion can lead to poor

This work is supported in part by NSF grants CCF-1740858, CCF-1712119 and DMS-1839274, DMS-1839307.

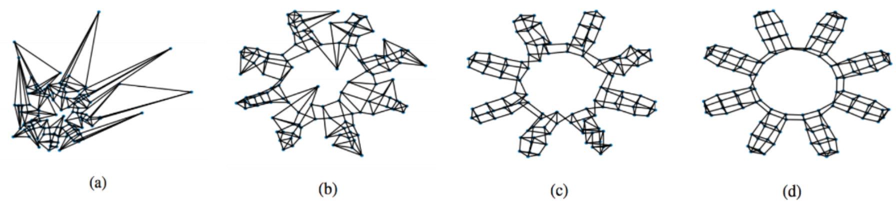


Fig. 1: Different layouts of the same graph from the crossing angle maximization Graph Drawing Contest: (a) from the Tübingen algorithm that won in 2018 [3]; (b) from the KIT algorithm that won in 2017 [8]; (c-d) from SPX with different balance in the optimization of stress, crossing angle, and edge crossings.

quality drawings [21]. As an extreme example, for minimum drawing area we can place all vertices on top of each other, yet perform poorly in the other quality criteria. A similar example is shown in Figure 1 where (a-b) show the outputs on a Graph Drawing Contest graph produced by two state-of-the-art algorithms for crossing angle maximization [3,8] while (c-d) show the outputs of SPX with different balance in the optimization of stress, crossing angle, and edge crossings. Note that the SPX approach better preserves topology and produces visually appealing results and although (d) has the lowest crossing angle, it arguably provides the most recognizable drawing. Delving further into this observation, we examined the contest graphs across several metrics, as shown in Figure 2, noting that optimizing for one criterion could yield extreme drawings. Graph 2018-8 in the middle row is a case where optimal crossing angle (center) requires a very large drawing area. Graph 2017-2 in the last row is a case where the best crossing angle (left) exhibits poor vertex resolution. These observations motivated us to seek a balance of criteria to improve drawings.

To demonstrate our framework, we formulate optimization terms for three criteria: minimizing edge crossings, maximizing the crossing angle, and upwardness (all of which have been used in Graph Drawing Contests). We compare our edge crossing formulation to state-of-the-art approaches on a corpus of community graphs. SPX achieves better edge crossing results than just optimizing stress, and frequently outperforms several of the crossing-centric algorithms. Similarly, we show that aiming only at the optimization of crossing angle tends to significantly impact the quality of the layout for other criteria. Although the angle-centric algorithms outperform SPX, our algorithm generates comparable crossing angle values and sometimes outperforms the angle-centric ones, while still achieving better performance on other drawing aspects. Finally, we compare our upwardness preserving approach to existing directed graph layout approaches [6,14,18,27].

In summary, our contributions are: (1) Stress-Plus-X (SPX), a framework for optimizing multiple graph drawing criteria simultaneously (Section 3); (2) Optimization terms for maximizing edge crossing angles (Section 3.2) and upwardness preservation (Section 3.3); and (3) An evaluation of our optimization

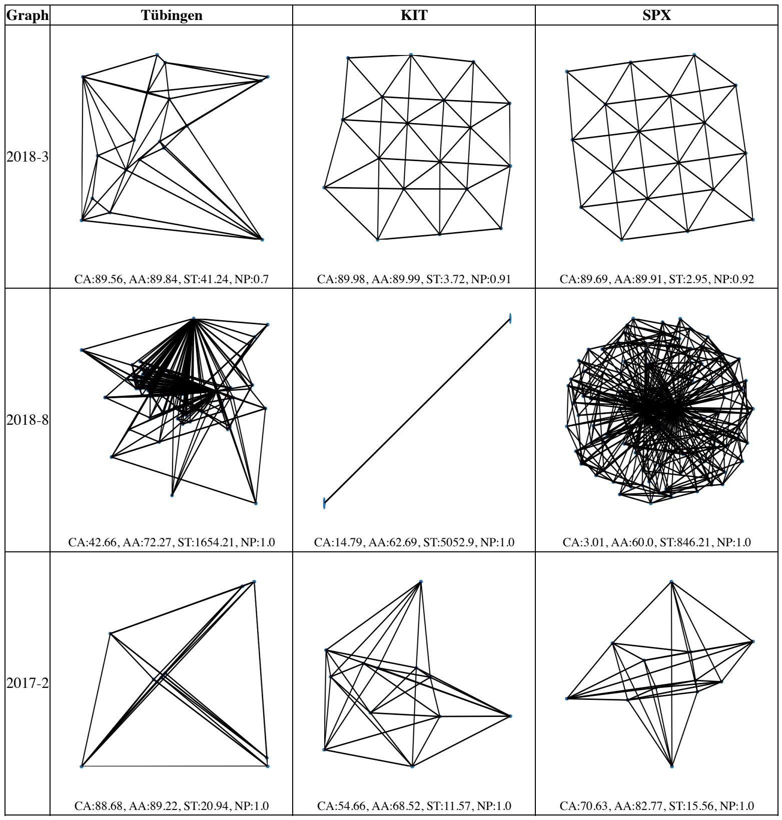


Fig. 2: Graphs from the 2017-18 Graph Drawing Contests. In graph 2018-3, the crossing angles are all within 1% of the optimal, yet SPX best shows the underlying graph structure. The best crossing angle layout for Graph 2018-8 (center) yields a large drawing area. The best crossing angle layout for Graph 2017-2 (left) yields poor vertex resolution. We report crossing angle (CA), average angle (AA), stress (ST), and neighborhood preservation (NP).

terms in comparison to state-of-the-art single criterion approaches (Section 4). An extended version of this paper is available at arxiv [11].

2 Background and Related Work

Existing graph layout algorithms usually optimize a single drawing criterion, e.g., minimizing stress or maximizing the minimum edge crossing angle. We define these criteria formally and discuss layout approaches that focus on them.

Stress: Stress measures the difference between node-pair distances in a layout and their graph-theoretic distances, based on an all-pairs shortest path computation. It is a natural measure of how well the layout captures the structure in the underlying graph. Let $\mathbf{C}_i$ be the position of the ith node in a layout $\mathbf{C}$ and d_{ij} be the graph distance between node pair i, j. Then stress($\mathbf{C}$) = $\sum_{i<j}(w_{ij}||\mathbf{C}_i - \mathbf{C}_j|| - d_{ij})^2$. A typical normalization value is $w_{ij} = d_{ij}^{-2}$.

Kamada and Kawai [22] formulate the graph layout problem as that of minimizing stress and use energy-based optimization. Gansner *et al.* [17] use stress majorization instead. Stress-based graph visualization can also be seen as a special case of a multi-dimensional scaling (MDS) [24,34], which is a powerful dimensionality reduction technique. Variants of MDS are used in many graph layout systems, including [5,17,28]. None of these methods aim to optimize other criteria such as minimizing edge crossings or maximizing crossing angles.

Wang *et al.* [36] reformulate stress to incorporate target edge directions and lengths and propose constraints to reduce crossings or improve crossing angle in given subgraphs, but not in the entire graph. Constrained layout algorithms [13,14] combine stress minimization or force-directed layout with separation constraints between node pairs. Constrained layouts, however, do not optimize for edge crossings or crossing angles. When used with force-directed layout algorithms (such as Fruchterman-Reingold [16]) instead of with stress minimization, stress is also not optimized.

Edge Crossings and Crossing Angles: Minimizing the number of crossings between edges in a graph layout has been shown to be an important heuristic in readability of graphs [29], prompting interest in several graph drawing contests [1,4]. Other than recent works by Radermacher *et al.* [30] and Shabbeer *et al.* [33] (discussed in Section 2), there is little work on directly minimizing edge crossings in general graphs.

The crossing angle of a straight-line drawing of a graph is the smallest angle between two crossing edges in the layout. Large crossing angles have been shown [2,20,21] to improve graph readability and several heuristics have been proposed to maximize crossing angles. Demel *et al.* (KIT) [8] propose a greedy heuristic to select the best position for a single vertex from a random set of points. Bekos *et al.* (Tübingen) [3] propose selecting a vertex arbitrarily from a set of vertices, called the *vertex-pool*, which contains a subset of the vertices which are adjacent to the pairs of edges that have the minimum crossing angle. Both approaches above performed very well in crossing angle maximization, but neither is concerned with stress minimization or other criteria.

Upward Drawing: A drawing of a directed acyclic graph is upward if the target vertex of each directed edge has a strictly higher y-coordinate than the source vertex. Upward drawing is used to show ordering or precedence between entities in a variety of settings [14,18]. Sugiyama layout [35] is the most common approach for creating upward drawings. The layout algorithm assigns ranks to the vertices to determine their y-coordinates followed by computing their x-coordinates to minimize crossings between consecutive layers. Examples include *dot* [18], *dagre* [6], and *OGDF* [27]. Mixed graphs, where only subgraphs are drawn upward, have also been drawn using this approach [32].

Neighborhood preservation and Drawing Area: While stress captures how well *global* graph distances are realized in the layout, neighborhood preservation captures how well *local* neighborhoods are preserved in the layout. This is the optimization goal of more recent dimensionality reduction techniques such as t-SNE [25] and UMap [26]. Specifically, in the context of graph drawing, neighborhood preservation is defined as the Jaccard similarity between the adjacent nodes in the graph and the nearest nodes in the layout, averaged over all nodes in the graph [23].

Drawing area refers to the size of the canvas used to layout the graph and is implicit when nodes are placed on an integer grid. Large drawing area is undesirable due to difficulties navigating the visualization or resolving the marks. Minimizing drawing area has also been used in Graph Drawing Contest challenges [12,19].

Joint Optimization: Our work aims to jointly optimize several graph drawing heuristics simultaneously. Huang *et al.* [21] previously optimized for two criteria simultaneously, namely crossing angle and angular resolution of the graph in a force-directed setting. Shabbeer *et al.* [33] minimized stress and edge crossings simultaneously using an optimization-based approach.

The objective function of Shabbeer *et al.* contains penalties for edge crossings. Edge crossings can be expressed as a system of non-linear constraints. Consider two edges $\mathbf{A} = \binom{a_1^x \; a_1^y}{a_2^x \; a_2^y}$ and $\mathbf{B} = \binom{b_1^x \; b_1^y}{b_2^x \; b_2^y}$ where the two nodes of $\mathbf{A}$ are (a_1^x, a_1^y), and (a_2^x, a_2^y) and similarly for $\mathbf{B}$. Farkas' Theorem can be used to state that the edges $\mathbf{A}$ and $\mathbf{B}$ do not cross if and only if there exists $\mathbf{u}$, and γ, such that

$$\mathbf{Au} + \gamma\mathbf{e} \geq \mathbf{0}, \mathbf{Bu} + (1+\gamma)\mathbf{e} \leq \mathbf{0} \tag{1}$$

where $\mathbf{e}$ is a 2-dimensional vector of ones. Intuitively, Eq.1 states that for a pair of edges $\mathbf{A}$ and $\mathbf{B}$ to not cross, there must exist a line that strictly separates the edges A and B, i.e., there is a non-zero margin between them. Here, $\mathbf{u}$ refers to a vector that is perpendicular to the direction of the separating line and γ is a scalar value that ensures the non-zero margin of separation between the edges.

This set of inequalities can be transformed into a penalty term, $penalty(\mathbf{A}, \mathbf{B})$, for edge pair $\mathbf{A}, \mathbf{B}$ such that it is zero for non-crossing edge pairs and strictly positive for crossing edge pairs.

$$penalty(\mathbf{A}, \mathbf{B}) = \min_{u,v} ||(-\mathbf{Au} - \gamma\mathbf{e})_+||_1 + ||(\mathbf{Bu} + (1+\gamma)\mathbf{e})_+||_1 \tag{2}$$

where $(z)_+ = max(0, z)$. The penalty term is combined with stress as a cost function and then iterative optimization is used to compute a layout. They demonstrate their approach on small biological networks.

Our approach differs in that our goal is a framework for balancing multiple criteria to achieve good results across them. We introduce penalties and constraints for crossing angle maximization and upward drawings. We further introduce a weighting to the edge crossings. Finally, we introduce a hyperparameter to directly balance across criteria.

3 SPX Algorithm

Stress-Plus-X (SPX) is a unified framework that can simultaneously optimize stress along with other graph drawing criteria. The "X" in SPX refers to the constraints that encode the additional criteria. We describe cost functions for encoding the number of edge crossings and crossing angle respectively, as well as constraints for preserving upwardness. The general SPX model is as follows:

$$cost(\mathbf{C}, \mathbf{u}, \gamma, \boldsymbol{\rho}) = stress(\mathbf{C}) + K \times \sum Penalties(\mathbf{C}, \mathbf{u}, \gamma, \mathbf{P}) \tag{3}$$

with node coordinates $\mathbf{C}$, balancing hyperparameter K, optional penalty parameters P (e.g., ρ_i in Section 3.1), and γ and $\mathbf{u}$ as described in Section 2.

Intuitively, decreasing stress, decreasing the penalty term for X, or decreasing both results in a decrease in the objective function. Hence, minimizing the objective function simultaneously optimizes for both stress and "X."

Modifying the value of K allows us to control the balance between the stress and the "X" terms. Figure 1 (c) and (d) show two layouts of the same graph created with different K parameterizations. Adjusting K to better balance criteria can result in a more intuitive drawing.

Optimization Procedure We optimize the cost function iteratively in two phases. We first compute the optimal $\mathbf{u}$ and γ for each pair of edges $(\mathbf{A}, \mathbf{B})$ via linear programming to minimize the penalties, $penalty(\mathbf{A}, \mathbf{B})$. Then, keeping the $\mathbf{u'}s$ and γ's constant, for all edge pairs, we optimize the cost function by modifying $\mathbf{C}$ using gradient descent; see Algorithm 1.

Algorithm 1 Stress-plux-X(G)

Compute initial layout $\mathbf{C}_0$ (using stress majorization, force-directed layout, or random initialization)
for Number-of-iterations **do**
  Keeping the node coordinates $\mathbf{C}$ constant, find optimal $\mathbf{u}$ and γ for each edge pair ($\mathbf{A}$,$\mathbf{B}$) using linear programming to minimize $penalty(\mathbf{A}, \mathbf{B})$
  Keeping $\mathbf{u}$'s and γ's constant, minimize $cost(\mathbf{C}, \mathbf{U}, \gamma, \boldsymbol{\rho})$ by updating $\mathbf{C}$ using gradient descent
end for

3.1 Stress plus Crossing Minimization

The edge crossings penalty is: $\sum_{i=1}^{l}(\rho_i/2)*\{||(-\mathbf{A}^i(\mathbf{C})u^i-\gamma^i\mathbf{e})_+||_1+||(\mathbf{B}^i(\mathbf{C})u^i+(1+\gamma^i)\mathbf{e})_+||_1\}$ where l is the number of edge pairs, $\mathbf{A}^i(\mathbf{C})$ and $\mathbf{B}^i(\mathbf{C})$ are the first and second edges of edge pair i as matrices $\mathbf{A}$ and $\mathbf{B}$, $\mathbf{u}^i, \gamma^i$ are the $\mathbf{u}, \gamma$ terms for edge pair i, and ρ_i is a weight on the penalty for edge pair i.

Shabbeer *et al.* [33] use a compounding weight where each edge crossing gets penalized more the longer it persists through the optimization iterations. We found that such a penalty can result in the introduction of new edge crossings for graphs that are larger and denser. With this in mind, we use a binary weight for ρ_i: the value is 1 when edges intersect and 0 otherwise.

The cost function for stress plus crossing minimization further differs from Shabbeer *et al.* in the criteria weighting parameter K. Figure 4 (Section 4) shows that the use of binary weights and hyperparameter K helps SPX achieve better results compared to Shabbeer *et al.*.

3.2 Stress plus Crossing Angle Maximization

Our crossing angle maximization penalty is the edge crossing penalty with an additional factor of $cos^2(\theta_i)$ in each factor of the summation, where θ_i is the angle between a pair of crossing edges. We use cos^2 to constrain to positive values and give a heavier weight to smaller crossing angles. Note this modified penalty function explicitly maximizes the minimum crossing angle and implicitly minimizes the number of crossings, as when a crossings is removed altogether it cannot contribute to the minimum crossing angle.

3.3 Stress plus Upward Crossing Minimization

We add the upwardness criteria to SPX by adding constraints to the model. Let (u, v) be a directed edge. Then, in the drawing of the graph the y coordinate of v should be strictly larger than the y coordinate of u. We enforce this directly with a linear constraint ($y_v > y_u$). If the input graph is a DAG then we add this constraint for all edges. If the graph is mixed then we add the constraints only for the directed edges.

3.4 Implementation

We implemented SPX in Python. It uses the stress majorization formulation of Gansner *et al.* [17] to minimize stress and the edge crossing detection code from Demel *et al.* [8]. SPX source code and experimental material are available at `https://github.com/devkotasabin/SPX-graph-layout`.

Initial Layouts We ran our experiments using 3 different layout algorithms as input to the SPX algorithm: stress majorization (`neato`), force-directed layout (`sfdp`), and random initialization. Both `neato` and `sfdp` are available in the

GraphViz package [15]. To ameliorate the effects of sensitivity to initial layout, we employ random starts of SPX, using each method multiple times and choosing the layout that maximizes the objective.

Gradient Descent Algorithms We experimented with the following algorithms for gradient descent (GD) [31]: `bfgs`, `l-bfgs`, vanilla GD, momentum-based GD, Nesterov momentum-based GD, `Adagrad`, `RMSprop`, and `Adam`. We found that for different types of graphs, different GD variants yielded better results and we kept all but `bfgs` and `l-bfgs` in our parameter sweep based on their performance in our pilot experiments. Section 3.5 contains further analysis of different GD variants and their convergence plots.

Parallelization Each combination of random initial layout, gradient descent algorithm, and value of K is independent and thus can be run in parallel. Operations on each edge pair, such as computing $\mathbf{u}$ and γ, as well as summing the penalties, can also be parallelized. However, running edge pairs fully in parallel would incur significant overhead. We leave the implementation of this approach as future work.

3.5 Convergence analysis

Figure 3 illustrates the convergence behavior of SPX using the six variants of gradient descent from Section 3.4 on two graphs, graph 5 from the community graphs of Section 4 (top row) and graph 9 from 2018 Graph Drawing contest (bottom row). Convergence behavior of the variants differ depending on graph. Figure 3 shows the values for number of crossings, stress, and crossing angle over 100 iterations for a fixed value of K(= 2) for both graphs.

For both graphs, at least one gradient descent variant converges within 100 iterations. In the first graph, `Momentum` and `Nesterov` converge rapidly and then get stuck in local minima. In the first graph, they overcome the local minima to continue convergence, while on the second graph they diverge after the minima. We hypothesize convergence per variant is dependent on graph properties and thus use all six. Further analysis and optimization is left for future work.

4 Results

As SPX is designed to be a flexible framework, we evaluate it in three different contexts. First, we compare SPX to Shabbeer *et al.* [33] on stress and number of crossings showing SPX performs better.

Second, we compare SPX to two state-of-the-art algorithms for crossing angle optimization: Demel *et al.* from KIT [8] and Bekos *et al.* from Tübingen [3]. We compare across five readability metrics discussed earlier: stress (ST), number of crossings (NC), crossing angle (CA), drawing area (DA), and neighborhood preservation (NP). We show SPX balanced multiple criteria simultaneously rather than optimizing one at the expense of others.

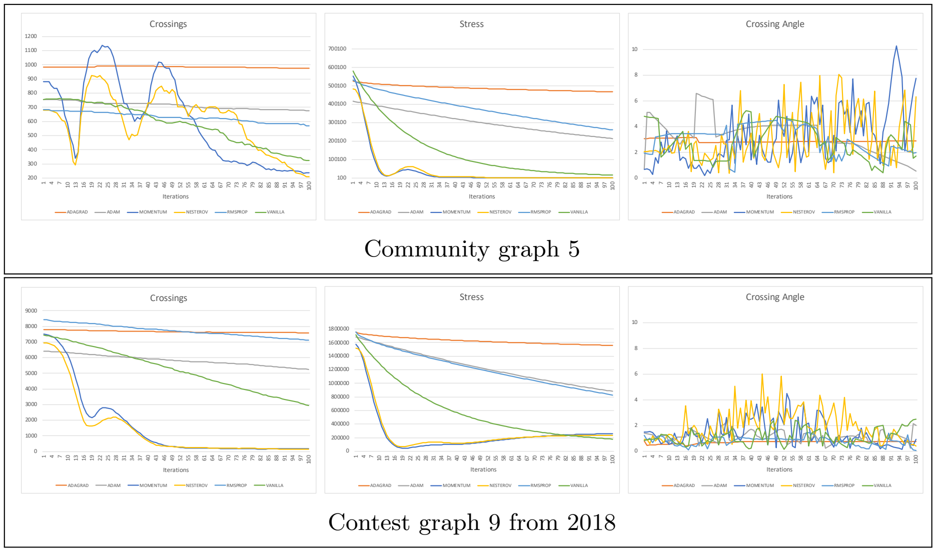


Fig. 3: Number of crossings, stress, and crossing angle over 100 iterations for 6 variants of GD algorithms on 2 graphs run with fixed K(= 2). The 2 graphs are graph 5 from random subset of 25 community graphs (top row) and graph 9 from 2018 graph drawing contest (bottom row).

Third, we compare SPX to existing approaches that directly optimize crossings [18,6,14,27] for upward drawings of DAGs. Our results show that SPX can preserve upwardness while performing better across other readability criteria.

4.1 Datasets and Experimental Settings

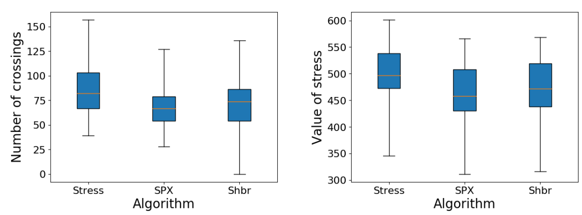
For the first two evaluations we used the 2017 and 2018 graph drawing contest graphs [9,10], as well as a collection of 400 graphs used in a crossings minimization study by Radermacher *et al.* [30]. For the third evaluation (upward drawing of DAGs) we generated 4 trees and 30 DAGs of different sizes.

We ran our experiments using all six gradient descent variants discussed in Section 3.4. We swept the values of K in the range of 2^{-5} to 2^{5} in exponential increments. We used three different initial layout algorithms as input: `neato`, `sfdp`, and random initialization with five different starts each. Metrics were calculated using the `graphmetrics` library of De Luca [7].

4.2 Comparison to Shabbeer *et al.*

We compare SPX with two algorithms - Shabbeer *et al.* [33] and stress majorization [17] on the corpus of 100 community graphs. The crossings value for stress is taken from Radermacher *et al.* [30] and the stress value calculated as lowest from five random `neato` [15] layouts. We run the SPX variant that performs stress-plus-crossing minimization only and compare using two metrics, number of crossings and stress, because Shabbeer *et al.* minimizes only for these two metrics. We do not perform the same two-metric comparison with the crossing minimization algorithms of Radermacher *et al.* [30] because they are not concerned with stress. We provide details about crossing minimization only for SPX and the algorithms of Radermacher *et al.*.

Figure 4a shows that on average SPX produces fewer crossings than both other approaches. Figure 4b shows that on average SPX produces layouts with lower stress than both other approaches. We hypothesize that SPX performs better than stress majorization on stress because of SPX's multiple random starts and the use of `neato` as one of the initializations.



(a) Crossing minimization (b) Stress minimization

Fig. 4: Comparing SPX, Shabbeer *et al.* and stress majorization in terms of the number of crossing and stress minimization using 100 community graphs.

4.3 Comparison Across Several Criteria

We examine several readability criteria across the layouts obtained by the three algorithms designed to minimize crossing angle: KIT [8], Tübingen [3], and SPX. In particular, we consider stress, neighborhood preservation, edge crossings, drawing area, and crossing angle.

Though our impetus was the graph drawing contest graphs, they are diverse in structure, making it difficult to compare across them. To perform a bulk comparison, we randomly select a subset of 25 graphs from community graphs described above.

Figure 5 shows the results for the 25 graphs, presented in a pairwise fashion of metrics. We plot the metrics so that points in the lower left corner indicate good performance in the two metrics. From the plots we can see that most of the SPX drawings are in the well-performing corner.

Figure 6 shows an example of a community graph, drawn by all three algorithms. SPX achieves best stress and crossing angle while performing very close to the winner, KIT, in terms of number of crossings.

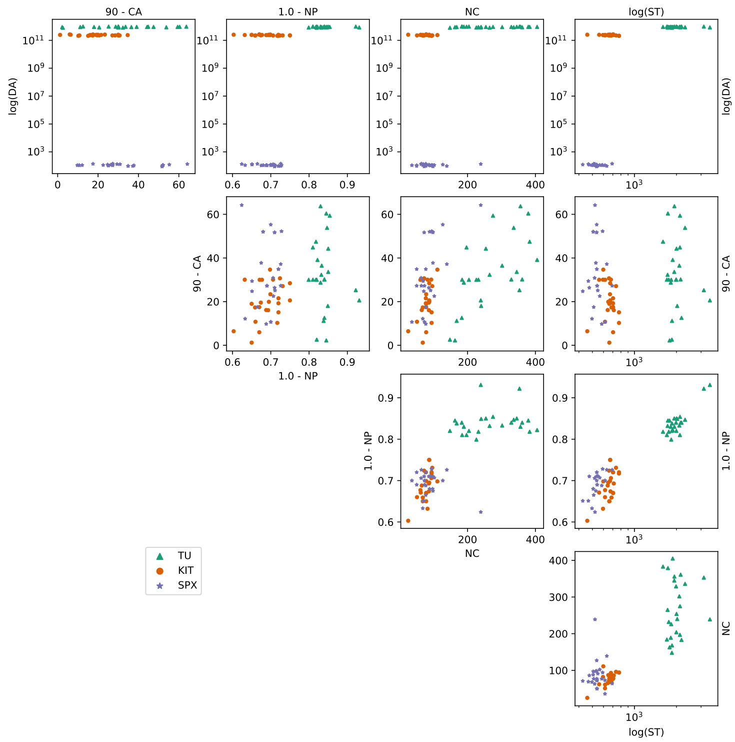


Fig. 5: Pairwise metric evaluation of the KIT, Tübingen, and SPX algorithms using stress (ST), number of crossings (NC), crossing angle (CA), neighborhood preservation (NP), and drawing area (DA).

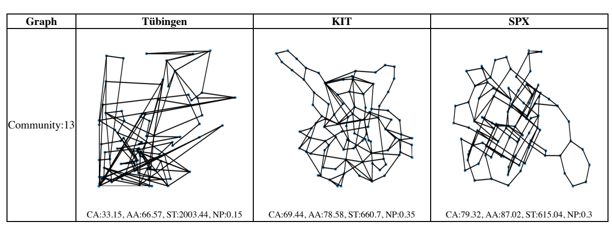


Fig. 6: Outputs of the Tübingen, KIT, SPX algorithms on a community graph.

4.4 Comparison of Upward Drawings

To evaluate SPX for upward drawing, we compare it to several state-of-the-art directed graph algorithms across several metrics on a corpus of 4 trees and 30 DAGs, described in [11].

We compared SPX to *dot* [18]; *dagre* [6] and both variants of Sugiyama in *OGDF* [27]: the barycenter heuristic ("*ogdfb*") and the median heuristic ("*ogdfm*"). We verified all algorithms, including SPX, produced completely upward drawings. We measured drawing area (A), stress (ST), and number of crossings (CR). We also measured height and width separately, but found their behavior to be the same as those for drawing area. The results of the experiment are reported in Table 1. Each cell indicates the number of times each algorithm had the best value for the metric, with ties being attributed to both algorithms.

	dagre	**dot**	**ogdfb**	**ogdfm**	**SPX**
ST	0	0	0	0	4
A	0	0	0	0	4
CR	4	4	4	4	4

4 binary trees

	dagre	**dot**	**ogdfb**	**ogdfm**	**SPX**
ST	0	0	0	0	30
A	0	0	0	0	30
CR	2	5	8	11	14

30 directed acyclic graphs

Table 1: The number of times each algorithm had the best metric value for upward drawings of 4 complete balanced binary trees (left) and 30 DAGs (right).

Table 1 shows that SPX consistently produces the best drawings across the metrics, although all other algorithms also produce planar layouts for the complete binary trees. However, there is a caveat in the measure of area. We do not impose any resolution to the upwardness of the drawings. The SPX drawings are very small in area compared to those generated by the other algorithms. Imposing a resolution constraint could increase crossings and stress, indicating a post-processing to enforce resolution may be a better option. We experimented with a naïve scaling parameter which results in very large area. We leave a more appropriate post-processing algorithm as future work.

5 Conclusions and Future Work

As some of the drawings in this paper show, optimizing just one layout criterion can result in unreadable drawings. It seems like a natural idea to consider approaches that balance multiple layout criteria. SPX is an example of such a graph layout framework that balances the optimization of multiple criteria and achieves quality that is close to one criterion state-of-the-art algorithms. Currently SPX considers stress minimization, crossing minimization, crossing angle maximization, and upwardness. A natural direction for future work is to incorporate additional layout criteria. Our current implementation of SPX relies on a combination of stress minimization and a linear program solver. As a result the algorithm is prohibitively slow for large graphs. Possible ways to speed up the algorithm, such as multi-level computation, are worth exploring.

References

1. 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). https://doi.org/10.1007/978-1-4614-0110-0_2
2. 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. GD'10, Springer-Verlag, Berlin, Heidelberg (2011), http://dl.acm.org/citation.cfm?id=1964371.1964378
3. 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. CoRR **abs/1808.10519** (2018), http://arxiv.org/abs/1808.10519
4. Buchheim, C., Chimani, M., Gutwenger, C., Jünger, M., Mutzel, P.: Crossings and planarization. Handbook of Graph Drawing and Visualization pp. 43–85 (2013)
5. Chen, L., Buja, A.: Local multidimensional scaling for nonlinear dimension reduction, graph drawing, and proximity analysis. Journal of the American Statistical Association **104**(485), 209–219 (2009)
6. Dagrejs: dagrejs/dagre, https://github.com/dagrejs/dagre/wiki
7. De Luca, F.: graphmetrics library. https://github.com/felicedeluca/graphmetrics (2019)
8. Demel, A., Dürrschnabel, D., Mchedlidze, T., Radermacher, M., Wulf, L.: A greedy heuristic for crossing-angle maximization. CoRR **abs/1807.09483** (2018)
9. Devanny, W., Kindermann, P., Löffler, M., Rutter, I.: Graph drawing contest report. In: International Symposium on Graph Drawing and Network Visualization. pp. 575–582. Springer (2017)
10. Devanny, W., Kindermann, P., Löffler, M., Rutter, I.: Graph drawing contest report. In: International Symposium on Graph Drawing and Network Visualization. pp. 609–617. Springer (2018)
11. Devkota, S., Ahmed, R., De Luca, F., Isaacs, K., Kobourov, S.: Stress-Plus-X (SPX) Graph Layout (2019)
12. Duncan, C.A., Gutwenger, C., Nachmanson, L., Sander, G.: Graph drawing contest report. In: International Symposium on Graph Drawing. pp. 575–579. Springer (2012)
13. Dwyer, T.: Scalable, versatile and simple constrained graph layout. Comput. Graph. Forum **28**, 991–998 (2009)
14. 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 (09 2006). https://doi.org/10.1109/TVCG.2006.156
15. Ellson, J., Gansner, E., Koutsofios, L., North, S.C., Woodhull, G.: Graphviz—open source graph drawing tools. In: International Symposium on Graph Drawing. pp. 483–484. Springer (2001)
16. Fruchterman, T.M.J., Reingold, E.M.: Graph drawing by force-directed placement. Software: Practice and Experience **21**(11), 1129–1164 (1991). https://doi.org/10.1002/spe.4380211102, https://onlinelibrary.wiley.com/doi/abs/10.1002/spe.4380211102
17. Gansner, E.R., Koren, Y., North, S.: Graph drawing by stress majorization. In: Pach, J. (ed.) Graph Drawing. pp. 239–250. Springer Berlin Heidelberg, Berlin, Heidelberg (2005)

18. Gansner, E.R., North, S.C., Vo, K.P.: Technique for drawing directed graphs (08 1990), uS Patent 4,953,106
19. Gutwenger, C., Löffler, M., Nachmanson, L., Rutter, I.: Graph drawing contest report. In: International Symposium on Graph Drawing. pp. 501–506. Springer (2014)
20. Huang, W., Eades, P., Hong, S.H.: Larger crossing angles make graphs easier to read. J. Vis. Lang. Comput. **25**(4), 452–465 (Aug 2014). https://doi.org/10.1016/j.jvlc.2014.03.001, `http://dx.doi.org/10.1016/j.jvlc.2014.03.001`
21. 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). https://doi.org/https://doi.org/10.1016/j.jvlc.2011.12.002, `http://www.sciencedirect.com/science/article/pii/S1045926X11000814`
22. Kamada, T., Kawai, S.: An algorithm for drawing general undirected graphs. Information Processing Letters **31**(1), 7 – 15 (1989). https://doi.org/https://doi.org/10.1016/0020-0190(89)90102-6, `http://www.sciencedirect.com/science/article/pii/0020019089901026`
23. 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 (Jun 2017). https://doi.org/10.1111/cgf.13187, `https://doi.org/10.1111/cgf.13187`
24. Kruskal, J.B.: Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. Psychometrika **29**(1), 1–27 (1964)
25. Maaten, L.v.d., Hinton, G.: Visualizing data using t-sne. Journal of machine learning research **9**(Nov), 2579–2605 (2008)
26. McInnes, L., Healy, J., Melville, J.: Umap: Uniform manifold approximation and projection for dimension reduction. arXiv preprint arXiv:1802.03426 (2018)
27. Mutzel, P., Chimani, M., Gutwenger, C., Klein, K.: Ogdf an open graph drawing framework. 15th International Symposium on Graph Drawing (02 2007). https://doi.org/10.17877/DE290R-7670
28. Pich, C.: Applications of multidimensional scaling to graph drawing. Ph.D. thesis (2009)
29. Purchase, H.: Which aesthetic has the greatest effect on human understanding? In: DiBattista, G. (ed.) Graph Drawing. pp. 248–261. Springer Berlin Heidelberg, Berlin, Heidelberg (1997)
30. Radermacher, M., Reichard, K., Rutter, I., Wagner, D.: A geometric heuristic for rectilinear crossing minimization. In: The 20th Workshop on Algorithm Engineering and Experiments (ALENEX'18). p. 129–138. Pagh, R., Venkatasubramanian, S. (eds.) (2018). https://doi.org/https://doi.org/10.1137/1.9781611975055.12
31. Ruder, S.: An overview of gradient descent optimization algorithms. CoRR **abs/1609.04747** (2016), `http://arxiv.org/abs/1609.04747`
32. Seemann, J.: Extending the sugiyama algorithm for drawing uml class diagrams: Towards automatic layout of object-oriented software diagrams. In: International Symposium on Graph Drawing. pp. 415–424. Springer (1997)
33. 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)
34. Shepard, R.N.: The analysis of proximities: multidimensional scaling with an unknown distance function. Psychometrika **27**(2), 125–140 (1962)
35. Sugiyama, K., Tagawa, S., Toda, M.: Methods for visual understanding of hierarchical system structures. IEEE Transactions on Systems, Man, and Cybernetics **11**(2), 109–125 (1981)

36. Wang, Y., Wang, Y., Sun, Y., Zhu, L., Lu, K., Fu, C., 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 (Jan 2018). https://doi.org/10.1109/TVCG.2017.2745919
37. Ware, C., Purchase, H., Colpoys, L., McGill, M.: Cognitive measurements of graph aesthetics. Information visualization **1**(2), 103–110 (2002)