FORBID: Fast Overlap Removal By stochastic gradIent Descent for Graph Drawing

Loann Giovannangeli, Frédéric Lalanne, Romain Giot, Romain Bourqui

HAL Id: hal-03774089
https://hal.science/hal-03774089

Submitted on 9 Sep 2022

FORBID: Fast Overlap Removal By stochastic gradIent Descent for Graph Drawing

Loann Giovannangeli[0000−0002−9395−6495], Frederic Lalanne[0000−0001−9108−0955], Romain Giot[0000−0002−0638−7504], and Romain Bourqui[0000−0002−1847−2589]

LaBRI, UMR CNRS 5800, University Bordeaux, 33405 Talence, France
{firstname}.{lastname}@u-bordeaux.fr

Abstract. While many graph drawing algorithms consider nodes as points, graph visualization tools often represent them as shapes. These shapes support the display of information such as labels or encode various data with size or color. However, they can create overlaps between nodes which hinder the exploration process by hiding parts of the information. It is therefore of utmost importance to remove these overlaps to improve graph visualization readability. If not handled by the layout process, Overlap Removal (OR) algorithms have been proposed as layout post-processing. As graph layouts usually convey information about their topology, it is important that OR algorithms preserve them as much as possible. We propose a novel algorithm that models OR as a joint stress and scaling optimization problem, and leverages efficient stochastic gradient descent. This approach is compared with state-of-the-art algorithms, and several quality metrics demonstrate its efficiency to quickly remove overlaps while retaining the initial layout structures.

Keywords: Layout adjustment · Overlap removal · Stress optimization · Stochastic gradient descent

1 Introduction

Most dimension reduction algorithms consider data as points (*e.g.*, Multi Dimensional Scaling [23, 24], Graph Layout [10, 16, 26]). However, most visualization tools represent them by shapes with an area, whether to encode additional data within the screen representation of the nodes, or because it is simply more visually pleasing for end-users. Rendering such data that was laid out as points with shapes creates *overlaps* that can severely hinder the representation readability by hiding information. If not handled directly inside the dimension reduction algorithm (*e.g.*, [11, 17]), it is then the responsibility of a post-processing *Overlap Removal* (OR) algorithm to remove these overlaps.

In this paper, we consider the OR problem in graph layouts context, meaning that our laid out data points are nodes positioned by a graph drawing algorithm. These nodes are represented as rectangles with a position and size in two dimensions. Other shapes can be considered as well, as long as it is possible to check for overlaps and measure a distance between them. We consider that a pair of nodes overlaps if the intersection of their shapes representations is not null.

An OR algorithm takes a set of nodes positions and sizes as input and move these nodes to remove *all* overlaps while optimizing two main criteria: *compactness* and *initial layout preservation*. An algorithm that uniformly upscales the initial layout until there is no overlap perfectly works (*e.g.*, uniform Scaling [4]), but its result is not satisfactory as it produces very sparse layouts where the nodes visible areas are significantly reduced. Hence, a good OR algorithm should be able to optimize *compactness* by preserving the scale of the initial layout, using its empty spaces to move nodes apart. Initial layout preservation is also of utmost importance in a graph drawing context since the initial graph layout is often computed to emphasize the graph structure. It is then imperative to preserve the mental map a user has of the initial layout. In addition, graph representations also include visualization of their edges. In that regard, optimizing compactness only is not suited to the graph context since it ends up hiding the graph edges.

The main contribution of this paper is FORBID[1]: Fast Overlap Removal By stochastic gradIent Descent, a novel OR algorithm dedicated to graph drawings that produces overlap-free layouts balancing compactness and initial layout preservation. To the best of our knowledge, it is the first method to explicitly optimize the conjunction of these two aspects. It models the problem as a *stress* function: each pair of nodes in the overlap-free layout should be put at an *ideal distance* such that (i) there is no longer overlaps in the layouts and (ii) the distances between all the nodes are preserved. This stress is then optimized with an efficient state-of-the-art stochastic gradient descent algorithm [26]. Leveraging Chen *et al.* [3, 4] evaluation protocol, FORBID is compared with major state-of-the-art OR algorithms on a set of quality metrics specifically selected for this purpose. It demonstrates great capabilities to preserve initial layouts while retaining a decent level of compactness.

The remainder of this paper is organized as follows. Section 2 presents related works principally centered around the description of OR algorithms and their evaluation as proposed in [3, 4]. Section 3 describes FORBID algorithm, while Section 4 reports its evaluation. Finally, Section 5 discusses visual examples of overlap-free layouts from several OR algorithms as well as FORBID convergence.

Notations: Let $G = (V, E)$ be a graph with $V = \{v_1, v_2, ..., v_N\}$ its set of $N = |V|$ nodes and $E \subseteq V \times V$ its set of edges. A graph layout is defined as a tensor $X \in \mathbb{R}^{N \times 2}$ where X_i is the node v_i projection in 2D. A node v_i is defined by a rectangle of width and height (w_i, h_i) centered in X_i. Two nodes overlap each other if the intersection between their rectangles is not null. For convenience, we define the set of overlapping pairs of nodes in a graph by $O \subseteq V \times V$. A corresponding overlap-free layout is defined as $X' \in \mathbb{R}^{N \times 2}$. The euclidean distance bewteen two nodes v_i and v_j is noted $||X_i - X_j||$.

2 Related Works

This section presents major prior works in the Overlap Removal (OR) field. As described in Chen *et al.* survey [3, 4], several efficient OR algorithms exist, many

---

[1]FORBID implementation: `https://github.com/LoannGio/FORBID`

of which (*e.g.*, [12, 14, 19]) rely on *scan line* [6] to detect overlap presency in $\mathcal{O}(N \log N)$ and find all overlaps in $\mathcal{O}(|O|N(\log N + |O|))$ which *can* be faster than the pairwise search in $\mathcal{O}(N^2)$. These OR algorithms focus on different contexts (*e.g.*, graph or generic 2D representation) and optimize different criteria (*e.g.*, compactness, layout preservation). PFS [19], PFS' [12], FTA [14], RWordle-L [22] and uniform scaling [3,4] rely on the *scan line* algorithm to remove overlaps. PFS, PFS' and FTA are made of two passes handling horizontal and vertical movements separately, while RWordle-L moves nodes on both axes at the same time. In the end, they all have a quadratic complexity according to how nodes movements are computed. PRISM [8] models OR as a stress optimization problem in a *proximity graph* (*i.e.*, Delaunay triangulation of the initial layout) of a layout and runs in $\mathcal{O}(t(mkN + N \log N))$ where m, k are optimization hyper-parameters and t depends on the number of overlaps. GTREE [20] leverages PRISM proximity graph to remove overlaps, but constructs a minimum spanning tree upon it to reduce the number of forces to compute. They both propose a good level of initial layout preservation. As FORBID idea is close to that of PRISM in some way, it will be further discussed in Section 3. VPSC [6,7] models OR as a set of constraints to relax but tends to highly deform the initial layout. Its complexity is $\mathcal{O}(CN \log C)$ where C is the number of constraints in $\mathcal{O}(N)$ to relax; leading to a final complexity in $\mathcal{O}(N^2 \log N)$. Finally, Diamond [18] is another constraint programming-based OR algorithm in $\mathcal{O}(N^2)$ that optimizes orthogonal order preservation. Its originality is to propose to temporarily rotate nodes by $45°$, representing them as *diamonds* to facilitate the constraints relaxations.
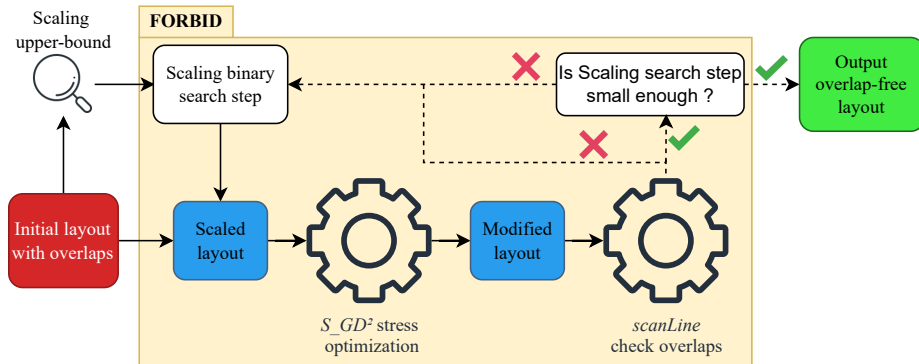
3 FORBID Algorithm

This section presents FORBID Overlap Removal (OR) algorithm. It is based on finding an optimal (*i.e.*, smallest) upscaling ratio while minimizing a stress function that models an overlap-free layout that preserves the initial one. The optimal scaling ratio is found with binary search, while the stress function is optimized with the *S_GD²* algorithm [26] that simulates stochastic gradient descent. An overview of the algorithm components is presented in Figure 1 and its complexity is in $\mathcal{O}(s(N^2 + N \log N))$ where s is defined later in Section 3.3.

3.1 Stress Modelization for Overlap Removal

Preliminaries. Traditional graph layout algorithms (*e.g.*, [2, 16, 21, 26]) often optimize a *stress* function that has been shown to lead to meaningful layouts and is defined as:

$$\sigma(X) = \sum_{i,j \in V} W_{ij}(||X_i - X_j|| - \delta_{ij})^2 \qquad (1)$$

where δ_{ij} is an *ideal distance* that the projected layout should preserve, and W_{ij} is a weight factor usually set to δ_{ij}^{-2}. In the graph layout context, δ_{ij} is set to

Fig. 1. Simplified schema of FORBID algorithm representing how the stress and upscaling criteria are organized together. Gears represent the algorithms FORBID relies on (*i.e.*, S_GD^2 [26] and *scan line* [6]), and colored boxes represent layouts while white boxes represent the search for the optimal scaling.

the graph theoretical distances so that the projected representation of the graph enables end-users to apprehend the graph structure.

As proposed by Gansner and Hu [8], stress is also a good criterion for OR problems. In fact, optimizing stress comes down to fit a distribution of distances in a low dimensional space to that of a higher space, considered ideal but of too high dimensions to be represented. The adaptation to an overlap removal context simply changes the notion of *ideal distances*. Rather than fitting the projected positions to match graph theoretical distances, the ideal distances are two-folded here. For a pair of nodes $p_{ij} = (v_i, v_j)$, with $p_{ij} \notin O$, the ideal distance δ_{ij} is set to their distance $||X_i - X_j||$ in the original layout. On the other hand, if $p_{ij} \in O$, δ_{ij} is set to a distance such that v_i and v_j do not overlap anymore; that distance depending on the nodes *shapes* and some design choices. This two-folded definition enables the optimization of both the preservation of the original layout and the overlapped nodes movement at the same time.

PRISM [8] is an example of OR algorithm that optimizes stress. It constructs a *proximity graph* (Delaunay triangulation of the initial layout) and optimizes stress alongside its edges. Considering nodes are represented as rectangles, they define the ideal distance as $\delta_{ij} = s_{ij}||X_i - X_j||$ where s_{ij} is an expansion factor of the edge (v_i, v_j) computed so that both nodes would be side by side. PRISM main limitations comes from the use of the *proximity graph* that does not capture all the overlaps and only enables the preservation of distances between close nodes in the initial layout, ignoring longer distances preservation.

FORBID Stress Modelization. For every pair of overlapped nodes, δ_{ij} is set to the distance between v_i and v_j centers if they were tangent in their corner. It allows one of them to be placed on the circle centered on the other and which radius is the minimum so that the nodes do not overlap anymore regardless of their

relative position. This distance ensures that two nodes do not overlap anymore and favors convergence by adding some margin space them, which is necessary for two reasons. First, since stress is optimized by stochastic gradient descent in FORBID (see Section 3.2), the ideal distances will most likely get approximated, but never perfectly matched. Second, the PRISM definition of ideal distance means there is only one correct placement for every pair of overlapping nodes that satisfies the stress. However, as opposed to PRISM, FORBID optimizes the distances between all N^2 pairs of nodes, making the number of constraints to relax much higher. This margin enables FORBID to converge faster toward a solution. Formally, the stress is defined as Equation 1 with δ_{ij} and W_{ij} set to:

$$\delta_{ij} = \begin{cases} \sqrt{\left(\frac{w_i+w_j}{2}\right)^2 + \left(\frac{h_i+h_j}{2}\right)^2}, \text{if } (v_i, v_j) \in O \\ \\ \|X_i - X_j\|, \text{if } (v_i, v_j) \notin O \end{cases} \tag{2}$$

$$W_{ij} = \begin{cases} \delta_{ij}^{k*\alpha}, \text{if } (v_i, v_j) \in O \\ \\ \delta_{ij}^{\alpha}, \text{if } (v_i, v_j) \notin O \end{cases} \tag{3}$$

where α is generally set to -2 and $k \in \mathbb{R}$ is an overlap-related factor to tailor the algorithm behavior to a desired initial layout preservation. The smaller the weight is given to overlapping pairs of nodes, the more the initial layout will be preserved at the cost of slower convergence or higher scaling.

3.2 FORBID Stress Optimization

FORBID optimizes stress by stochastic gradient descent by leveraging S_GD^2 [26] algorithm. S_GD^2 models stress as a set of constraints that are relaxed by individually moving pairs of nodes. The process is based on constrained graph layout [1, 5], and is optimized by considering the constraints individually to efficiently model clothes movements [15]. By individually moving pairs of nodes to optimize the stress, the algorithm can create new overlaps. And since overlapping and non-overlapping pairs of nodes have different notions of ideal distances, δ_{ij} and W_{ij} are updated at each optimization iteration (everytime all pairs of nodes have been moved once) so that, at any time, the algorithm optimizes distances according to the current state of the layout.

S_GD^2 optimization convergence is based on an annealing step size schedule that mimics a stochastic gradient descent. It computes $\mathcal{O}(N^2)$ movements, but in practice it converges in very few iterations, making it competitive with other algorithms (see Section 4.3). To make it even faster, we also stop the gradient descent if, at any iteration, the sum of all nodes movements in the current iteration is null. This explains why every execution of S_GD^2 is not necessarily of the same length (see Section 5).

3.3 Scaling to Ensure Convergence

Here, we define the *bounding box* as the minimum rectangle in which the initial layout fits. Removing all overlaps in a layout is sometimes not feasible without deforming its bounding box, *e.g.*, when the sum of the nodes areas exceeds the layout bounding box area. Two strategies can overcome this: (i) to allow the optimization algorithm to distort the bounding shape, or (ii) to scale up the drawing to create empty spaces that can be used to move nodes. These two choices boil down to considering that either "more space is needed" or "nodes must be smaller" to provide an overlap-free layout. The first strategy has the benefit of limiting the bounding box upscaling, but can result in strongly distorted layouts that makes it difficult to recognize the original graph structure. To guarantee that FORBID finds a solution, it uses the second method and searches for the optimal upscale ratio so that there is enough space to remove overlaps without deforming its aspect. The optimal upscale ratio s_{max} is found by binary search between 1 and the minimum scaling ratio for an overlap-free layout that has not moved any node (*i.e.*, scaling ratio of Scaling [4]). FORBID moves the nodes until there is no overlap (*scan line* [6]) *and* until the scaling ratio is optimal up to a given precision s_{step} (*e.g.*, 0.1).

We name *pass* a call to the optimization algorithm S_GD^2 (see Section 3.2) and *iteration* a step within S_GD^2 (*i.e.*, moving all pairs of nodes once). The maximum number of passes over S_GD^2 is defined by the binary search maximum depth $s \leq \log\left(\frac{s_{max}-1}{s_{step}}\right)$. Since the pass in S_GD^2 costs $\mathcal{O}(N^2)$ and that we test if there remains any overlap right after with the *scan line* [6] algorithm that executes in $\mathcal{O}(N \log N)$; the final complexity of FORBID is $\mathcal{O}(s(N^2 + N \log N))$.

As every pass in the optimization algorithm resets the annealing step size schedule (see Section 3.2), the nodes can be moved a lot at the beginning of every pass; meaning that FORBID is somewhat allowed to modify the initial layout more than expected. Hence, we also experiment a variant of FORBID, called FORBID', in which the model starts from the scaled initial layout at every pass of the optimization algorithm. We expect this variant to be able to preserve the initial layout even better, probably at the cost of convergence speed.

Algorithm 1 presents FORBID algorithm pseudo-code. It has been simplified by removing the first pass that occurs with the initial scaling of the layout if the sum of nodes areas is lower than the layout bounding box. In practice, it only enters the while loop (line 20) to search for the optimal scale *if* scaling is necessary. The only modification required to implement FORBID' is to change line 21 into $X' \leftarrow$ scaleLayout($X, curScale$) so that the next pass in the optimization algorithm starts with the scaled initial layout.

4 FORBID Evaluation

For the sake of reproducibility, we used the same evaluation protocol as in the Chen *et al.* survey [3,4]. That includes quality metrics, datasets and algorithms. This section describes this protocol and presents the results of FORBID comparison

Algorithm 1 FORBID pseudo-code

1: **Methods**
2:    getScalingRatio($Layout$, $Sizes$): returns the minimum scaling ratio [3,4] so that there is no overlap anymore in $Layout$
3:    containsOverlap($Layout$, $Sizes$): [6] returns $true$ if there is overlap in $Layout$, $false$ otherwise
4:    scaleLayout($Layout$, $scaleFactor$): returns $Layout$ scaled by $scaleFactor$
5:    SGD2_StressOPT($Layout'$, $Layout$, $Sizes$): One pass of S_GD^2 [26] to optimize overlap removal modeled as stress
6: **end Methods**
7:
8: **Variables**
9:    X: Initial layout, 2D position of every node
10:    S: Nodes sizes in 2D, $S_i = (w_i, h_i)$
11:    $scaleStep$: Scaling step size to stop the search of optimal scale
12: **end Variables**
13:
14: **procedure** FORBID(X, S, $scaleStep$, $SGD2\_HP$)
15:    $lowScale \leftarrow 1$
16:    $upScale \leftarrow$ getScalingRatio(X, S)
17:    $curScale \leftarrow (lowScale + upScale)/2$
18:    $X' \leftarrow X$
19:    $thereIsOverlap \leftarrow$ containsOverlap(X', S)
20:    **while** $thereIsOverlap$ **or** ($upScale - lowScale > scaleStep$) **do**
21:       $X' \leftarrow$ scaleLayout(X', $curScale$)
22:       $X' \leftarrow$ SGD2_StressOPT(X', X, S)
23:       $thereIsOverlap \leftarrow$ containsOverlap(X', S)
24:       **if** $thereIsOverlap$ **then**
25:          $lowScale \leftarrow curScale$
26:       **else**
27:          $upScale \leftarrow curScale$
28:       **end if**
29:       $curScale \leftarrow (lowScale + upScale)/2$
30:    **end while**
31:    **return** X'
32: **end procedure**

---

with the selected algorithms on these datasets and metrics. Finally, FORBID execution time is also compared with some algorithms.

4.1 Evaluation Protocol

Quality Metrics. The quality metrics used to compare FORBID with other algorithms from the literature were selected by Chen *et al.* [3,4]. All are oriented as *lower is better*, the optimal value being 0 unless specified otherwise.

    oo_nni: stands for the *Orthogonal Ordering: Normalized Number of Inversions* and counts the number of times the nodes orthogonal order have been violated.

sp_ch_a [22]: is for *Spread Minimization: Convex Hull Area*. It measures by how much the convex hull area of the overlap-free layout is different from the one of the initial layout: $sp\_ch\_a = \frac{\text{convex\_hull\_area}(X')}{\text{convex\_hull\_area}(X)}$, the optimal value being 1. This metric mainly measures the layout scaling.

gs_bb_iar: means *Global Shape preservation: Bounding Box Improved Aspect Ratio* and is a variant of the aspect ratio between the bounding box of the initial and overlap-free layouts in which the minimal and target value is 1.

nm_dm_imse: stands for *Node Movement minimization: Distance Moved Improved Mean Squared Error*. It quantifies how much the nodes moved from their position in the initial layout to theirs in the overlap-free layout. To lessen the effect of positions value domains, the layouts are aligned as follows:

$$\text{nm\_dm\_imse} = \frac{1}{N} \sum_{v_i \in V} ||X'_i - \text{scale}(\text{shift}(X_i))||^2 \tag{4}$$

where *shift* and *scale* are moving and scaling the initial layout bounding box to match the center and dimensions of the overlap-free layout. It is important to mention that both in [3, 4] and therefore in this paper, the bounding boxes computed to scale the layouts do not take the nodes sizes into account.

el_rsd: is for *Edge Length preservation: Relative Standard Deviation*. It measures by how much the lengths of edges in the Delaunay Triangulation graph of an initial layout are preserved, *i.e.*, how well short-distances are preserved.

Datasets. Still following Chen *et al.* [3, 4] evaluation protocol, we use the Generated and Graphviz datasets available online[2].

Generated is a set of 840 synthetic graphs specifically generated for the benchmark in [3, 4]. It is made of 120 graphs of each size $10, 20, 50, 100, 200, 500, 1000$, laid out with the FM^3 algorithm [10]. These layouts have $2770 \pm 7567(std)$ initial overlaps in average, ranging between 0 and 31843.

Graphviz is a set of 14 real-world graphs from the Graphviz suite. They have between 36 and 1463 nodes and are laid out with $SFDP$ algorithm [13] and have for between 4 and 11582 initial overlaps ($2118 \pm 4078(std)$ in average).

Baseline Algorithms. As already state (see Section 2), there are two main criteria to optimize in overlap removal algorithms: compactness and initial layout preservation. By design, FORBID belongs to the second category and is then only compared with its corresponding algorithms (*i.e.*, PFS [19], PFS' [12], PRISM [8], GTREE [20] and Diamond [18]). Other algorithms create embeddings so compact that it is not even possible to visualize the graph edges and structures anymore; meaning they are not suited to overlap removal for graph visualization. *Scaling* is also excluded since it does not look for balance and rawly upscales the layout; which is not a satisfactory solution on its own.

---

[2]Generated and Graphviz graphs: `https://github.com/agorajs/agora-dataset`, last consulted on May 2022

4.2 Comparison with Baseline Algorithms on Quality Metrics

This section reports and discusses the performances of FORBID and the selected algorithms from the literature on the **Generated** and **Graphviz** datasets.

On the **Generated** dataset (see Figure 2), every algorithm succeeds in minimizing oo_nni, with Diamond and FORBID having slightly higher scores than others on worst cases (Q3). On sp_ch_a and gs_bb_iar, FORBID and FORBID' have the best scores by a fair margin, especially on worst cases (Q3). This demonstrates a good capability to limit the upscaling of the drawing in complicated layouts. Both FORBID and FORBID' also minimize nodes movements nm_dm_imse more than other algorithms, especially FORBID' that barely moves the nodes even on complex cases ($nm\_dm\_imse = 50.23$ on Q3). Finally, FORBID and its variant both have high scores on el_rsd. As defined in Section 4.1, this metric measures the edge length preservation along the edges of the Delaunay Triangulation (DT) of a graph. Hence, it does not measures that the lengths of the actual graph edges are preserved, but rather quantifies the preservation of the distances between the closest nodes. On the other hand, FORBID focuses on the preservation of *all* the nodes pairwise distances. These two strategies do not have the same notion of *preservation* of the initial layout and by ignoring long distances, PRISM tends to break the overall layout aspect. In addition, since overlapped nodes are likely

		FORBID	FORBID'	PFS	PFS'	PRISM	GTREE	Diamond
	Q1	0.00	0.00	0.00	0.00	0.00	0.00	0.04
oo_nni	Median	0.00	0.00	0.00	0.00	0.01	0.02	0.06
	Q3	0.06	0.03	0.00	0.00	0.02	0.03	0.09
	Q1	1.00	1.00	1.04	1.00	1.01	1.12	1.16
sp_ch_a	Median	1.03	1.03	1.69	1.22	1.12	1.49	1.96
	Q3	2.86	3.21	17.63	5.04	4.22	5.94	6.94
	Q1	1.00	1.00	1.01	1.00	1.00	1.01	1.04
gs_bb_iar	Median	1.01	1.00	1.19	1.04	1.04	1.04	1.07
	Q3	1.04	1.01	1.65	1.14	1.23	1.08	1.12
	Q1	3.90	3.90	6.50	1.65	9.82	28.22	535.65
nm_dm_imse	Median	34.69	25.03	694.83	116.06	131.57	292.56	1971.15
	Q3	419.88	50.23	7899.2	1782.80	688.1	2221.70	16571.10
	Q1	0.11	0.11	0.03	0.02	0.04	0.05	0.23
el_rsd	Median	0.37	0.32	0.19	0.11	0.16	0.16	0.66
	Q3	0.70	0.46	0.25	0.18	0.21	0.21	0.99

Fig. 2. Quality metrics quartile values for each algorithm on the **Generated** dataset (see Sect. 4.1). Cells color are selected based on the median value of the algorithms on each metric to enhance comparisons readability. The greener/lighter a cell color is, the better its quality metric score.

	FORBID	FORBID'	PFS	PFS'	PRISM	GTREE	Diamond
oo_nni	0.05	0.02	0.00	0.00	0.02	0.02	0.05
sp_ch_a	3.71	5.39	210.28	6.92	2.18	4.02	30.63
gs_bb_iar	1.05	1.02	1.97	1.22	1.33	1.17	1.10
nm_dm_imse	43795.43	2697.03	9768157.94	63594.74	42919.66	37331.83	1348426.00
el_rsd	0.55	0.42	0.34	0.22	0.28	0.26	0.23

Fig. 3. Quality metrics mean values on the **Graphviz** dataset (see Sect. 4.1). The greener/lighter a cell color is, the better its quality metric score.

to be adjacent in the DT graph and since our ideal distance between them is not the shortest possible (see Section 3.1), it was expected that FORBID would distort these distances more than other algorithms (*e.g.*, PRISM).

The performances on the **Graphviz** dataset are reported in Figure 3. On that dataset, the same trend can be observed with slight differences. Here again, every algorithm successfully minimizes *oo_nni* and both FORBID and FORBID' are still ahead by a fair margin on *gs_bb_iar*. However, they are no longer the bests on *sp_ch_a* and FORBID' is the only one to keep the lead on *nm_dm_imse*. On those two metrics, PRISM is slightly better than FORBID, while FORBID' has a deteriorated *sp_ch_a* for a much better node movement *nm_dm_imse* in comparison to other algorithms and to its own performances on the **Generated** dataset. In fact, since FORBID' focuses on preserving the initial layout, it tends to upscale the layouts more (worsening *sp_ch_a*) to minimize the nodes movements (improving *nm_dm_imse*). Finally, both have again the highest *el_rsd* scores, though the difference with other algorithms is less pronounced.

4.3   Execution Time Comparison with some Baseline Algorithms

This section compares FORBID execution time with that of the best techniques according to the performances observed in Section 4.2: PFS', GTREE and PRISM.

Execution times on the **Graphviz** dataset are reported in Figure 4 and enable to categorize three groups of difficulty. FORBID, FORBID' and PFS' are instantaneous on *easy* graphs, taking less than 20*ms* to solve the OR problem, being about ten to twenty times faster than GTREE and PRISM. On *medium* complexity graphs, PFS' achieves again the best performances. FORBID is slower, but remains faster than GTREE and PRISM by a fair margin, while FORBID' is slower than GTREE. On these graphs, PRISM is significantly slower than the

	$|V|$	$|E|$	$|O|$	Graph	FORBID	FORBID'	PFS'	GTREE	PRISM
Easy	36	107	4	dpd	0	0	0	29	34
	41	49	20	unix	0	0	0	46	54
	43	64	9	rowe	0	0	0	41	46
	47	55	33	size	1	1	0	49	91
	50	99	13	ngk104	0	0	0	40	46
	76	104	19	nan	1	1	0	58	115
	79	181	33	b124	4	5	0	77	229
	135	366	53	b143	13	19	0	126	275
Medium	213	269	1105	mode	166	420	1	270	1040
	302	611	268	xx	305	682	2	540	1450
	302	611	282	b102	268	824	2	540	1750
Hard	1054	1083	11582	root	3097	12989	22	7256	37734
	1235	1616	10540	badvoro	4577	18013	34	9694	27514
	1463	5806	5691	b100	11827	22880	48	13074	63877

Fig. 4. Execution time (in ms) of FORBID, FORBID', GTREE and PRISM on the **Graphviz** dataset (see Sect. 4.1). Rows are sorted by "graph complexity" measured with their number of nodes |V|, edges |E| and overlaps |O|. The greener/lighter a cell color is, the better its quality metric score.

four others. Finally, on the *hard*est graphs, PFS' is still almost instantaneous. FORBID is faster than the remaining methods, while FORBID' loses to GTREE. On *hard* graphs, PRISM becomes dramatically slower than other algorithms.

In the end, we can conclude that FORBID scales well with the OR problem complexity, while it is more difficult for FORBID'. For this last, the constraint to start from the scaled initial layout at each pass in the optimization algorithm significantly slows its convergence (see Section 5) and is responsible of the higher upscaling observed in Section 4.2 that enables a better layout preservation. In comparison with PRISM, which also optimizes OR modeled as stress, both FORBID variants are much faster, while PFS' faster handles large graphs.

5   Discussion

This section discusses FORBID behavior on some specific graph examples by comparing it to other algorithms. It also briefly discusses the convergence of FORBID and FORBID' variants, highlighting their different behaviors.
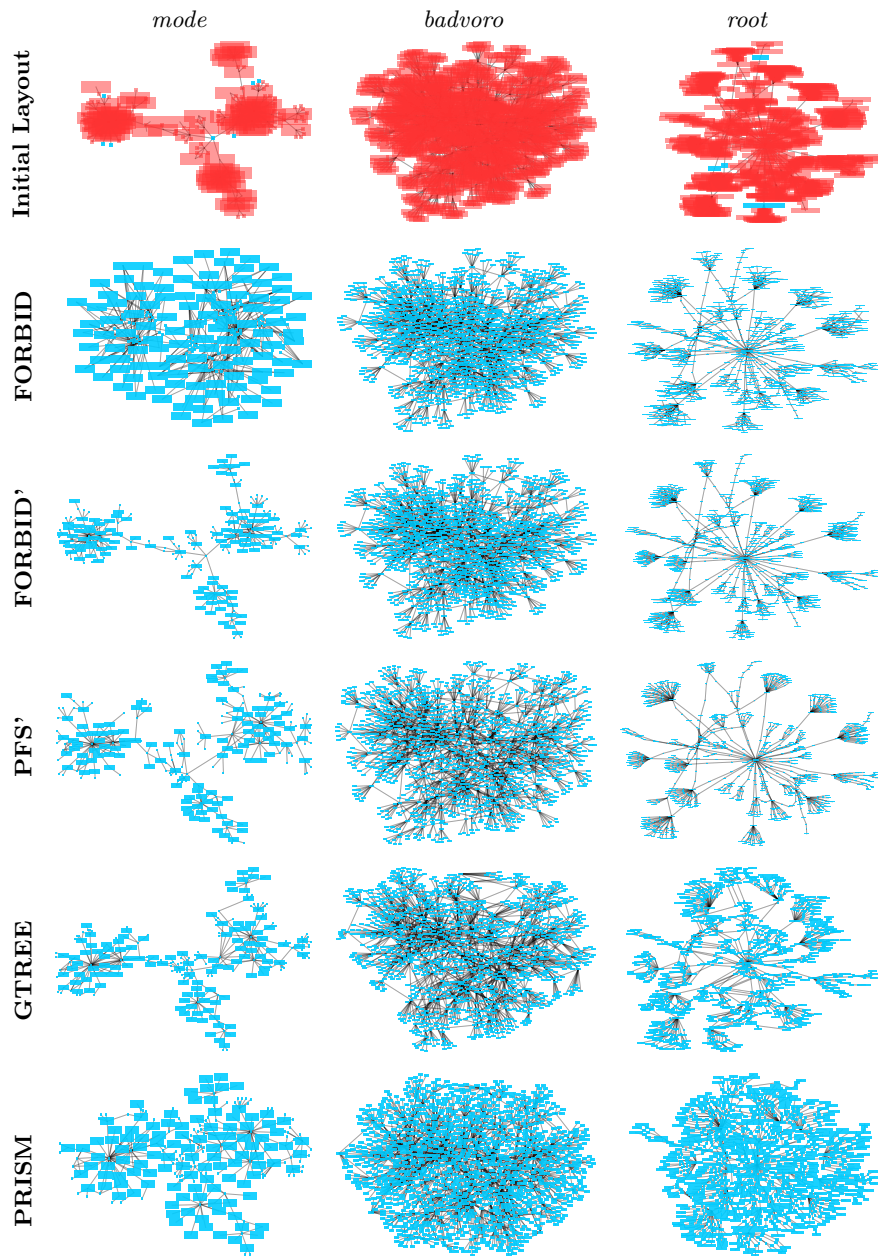
Visual Evaluation. This section focuses on the study of FORBID, FORBID', PFS', GTREE and PRISM on three graphs of the **Graphviz** dataset: *mode*, *badvoro* and *root*. Both initial and overlap-free layouts are presented in Figure 5, while the methods quality metrics are reported in Table 1.

On *mode*, both FORBID and PRISM damaged the initial layout to produce more compact embeddings. On the other hand, the others preserved the initial layout structure, FORBID' being the most pleasing one. The quality metrics corroborate this observation, with FORBID and PRISM having lower sp_ch_a (*i.e.*, upscaling) scores while FORBID', PFS' and GTREE have smaller nm_dm_imse (*i.e.*, nodes movements) scores; FORBID' having the smallest one.

On both *badvoro* and *root*, the same trend can be observed between the Overlap Removal methods. PRISM produces more compact embeddings that makes it difficult to recognize the initial layout and even to visualize edges. On the other hand, GTREE produces less compact layouts, but distorts the initial graph structures. Finally, FORBID, FORBID' and PFS' are satisfactory, but PFS' upscales the initial layout more than necessary. Again, these observations are corroborated by the quality metrics presented in Table 1: PRISM is consistently better on sp_ch_a, GTREE is not the best on any metric, and the remaining three have less nodes movements but higher upscaling. The high nm_dm_imse scores of PFS' can be imputed to the fact that this metric is sensitive to the overlap-free layout scale, even though the graph layout structures seem visually preserved.

Overall, FORBID and FORBID' produce balanced layout that optimize both the initial layout preservation and the embedding compactness.

Convergence Analysis. This section studies how re-initializing the layout at every *pass* (*i.e.*, call to the optimization algorithm) in FORBID' affects its convergence speed in comparison to FORBID. Observations of both variants on

Fig. 5. Graph visualization of examples from the **Graphviz** dataset with FOR-BID, FORBID', PFS' GTREE and PRISM. Nodes are colored in transparent red if they overlap each other, and opaque blue otherwise.
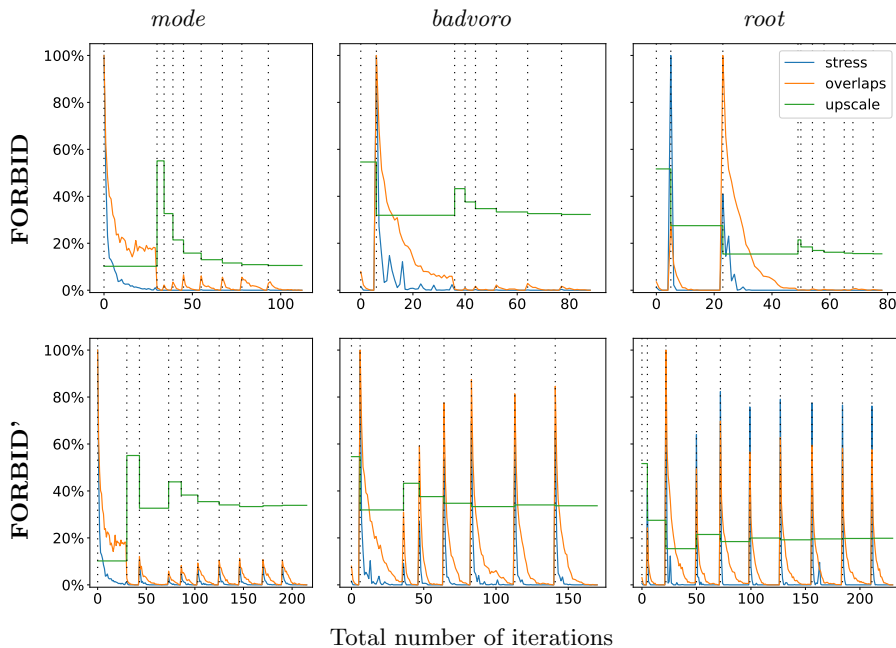
Table 1. Quality metrics of the drawings presented in Figure 5. For each graph and quality metric, the best score is highlighted in bold.

		FORBID	FORBID'	PFS'	GTREE	PRISM
mode	*oo_nni*	0.20	**0.03**	0.54	0.05	0.06
	sp_ch_a	**2.98**	10.35	9.08	7.77	3.72
	gs_bb_iar	1.02	**1.01**	1.20	1.23	1.52
	nm_dm_imse	20417	**1657**	8655	10238	17442
	el_rsd	0.95	0.70	**0.39**	0.54	0.62
badvoro	*oo_nni*	0.01	**0.00**	0.48	0.02	0.02
	sp_ch_a	11.42	12.40	17.92	13.45	**6.87**
	gs_bb_iar	**1.00**	**1.00**	1.25	1.17	1.85
	nm_dm_imse	979	**451**	57267	67296	47708
	el_rsd	0.32	0.23	**0.20**	0.40	0.39
root	*oo_nni*	**0.01**	**0.01**	0.53	0.05	0.04
	sp_ch_a	19.17	29.99	48.34	14.29	**6.27**
	gs_bb_iar	**1.01**	**1.01**	1.34	1.30	2.04
	nm_dm_imse	24502	**12151**	662185	356632	450287
	el_rsd	0.90	0.72	**0.43**	0.67	0.72

the *mode*, *badvoro* and *root* **Graphviz** graphs are presented in Figure 6. Each plot reports the number of passes, the length of each pass and the stress, number of overlaps and scaling ratio evolution against the total number of iterations.

For both FORBID and FORBID', the number of passes never exceeded 10. In these executions, the maximum number of iterations in the optimization algorithm was set to 30. Many passes stop before reaching this limit thanks to the stop condition on null nodes movements (see Section 3.2). As the stress and number of overlaps follow the same trends on every plot, it confirms that optimizing our stress effectively removes overlaps. The difference between FORBID and FORBID' is also distinctly observable. In FORBID, most overlaps are removed in the first few passes while the last ones are dedicated to the search for the optimal upscaling ratio while preserving the overlap-free layout. On the other hand, FORBID' has to restart from the scaled initial nodes every time a new pass begins, the problem being made simpler or harder through a different upscaling ratio. This explains why FORBID' is consistently slower than FORBID (see Section 4.3) but better preserves the initial layout (see Section 4.2).

Finally, we would like to discuss the choice of the binary search upper-bound s_{max}. In this paper, it was set to the minimum scaling ratio for an overlap-free layout that has not moved any node (see Section 3.3). Doing so guarantees that FORBID finds a overlap-free layout and solve the Overlap Removal task. Such an s_{max} value can be high when two nodes are almost perfectly overlapped, but in practice FORBID output layouts scaling remained far from this upper-bound during the benchmark (*e.g.*, green lines in Figure 6). Nevertheless, it would be possible to set s_{max} to a lower value to enforce a smaller scaling in the produced layout, at the cost of initial layout preservation and the guarantee that it is overlap-free. For instance, s_{max} could be set to twice the sum of nodes areas,

Fig. 6. Convergence plots of FORBID and FORBID'. It reports the evolution of stress, number of overlaps and scaling ratio against the total number of iterations. Vertical dashed lines represent beginning of new *passes*. Stress and number of overlaps are normalized by their respective maximum value, while upscaling ratio is normalized by its binary search maximum bound (see Section 3.3).

leading toward a layout which bounding box space is at least half occupied, but there would be no guarantee that this restrained space is enough to provide an overlap-free layout that preserves the initial one. The notion of *overlap* itself could also be approximated with a *tolerance* to speed up the convergence (*i.e.*, consider as *not* overlapped the pairs of nodes that do overlap by less than a *tolerance* margin). In the end, these choices mainly depend on the initial graph layout and the desired aspect of overlap-free layout. Although it can be slightly suboptimal, the s_{max} defined and used in this paper (see Section 3.3) is better adapted to the general case.

6 Conclusion

This paper has presented FORBID, an Overlap Removal (OR) algorithm that leverages upscaling and stress optimization by simulated stochastic gradient descent to minimize deformations in the initial layout. FORBID idea is based on combining upscaling and the preservation of all nodes pairwise distances to produce an overlap-free layout, focusing on the preservation of the initial

graph layout structures while limiting the surface used. It has been compared to several state-of-the-art algorithms, and is among the best techniques to preserve the initial layout, which is critical in graph drawing to retain the readability of the graph layout structures. FORBID complexity is in $\mathcal{O}(s(N^2 + N \log N))$ and is among the fastest methods on the benchmark graphs (with up to 1463 nodes and 11 582 Overlaps). Future works leads include improvements of the algorithm complexity to better handle large graphs. The first idea to achieve that is to sub-sample the nodes pairs to process to remove overlaps. A multi-scale approach could enable to optimize the initial graph layout structures preservation while sampling the distances to preserve (*i.e.*, preserve distances between-clusters and within-cluster; ignore between nodes of different clusters). Finally, with the recent advances in Deep Learning for graph drawing [9,25], we plan to learn a Deep Learning model solve OR problems. By design, these models can scale to large graphs as they are capable of solving the task they have learned in almost constant time.

Acknoledgements

References

1. Bostock, Michael and Ogievetsky, Vadim and Heer, Jeffrey: D^3 data-driven documents. IEEE transactions on visualization and computer graphics **17**(12), 2301–2309 (2011)
2. Brandes, Ulrik and Pich, Christian: An experimental study on distance-based graph drawing. In: International Symposium on Graph Drawing. pp. 218–229. Springer (2008)
3. Chen, Fati and Piccinini, Laurent and Poncelet, Pascal and Sallaberry, Arnaud: Node overlap removal algorithms: A comparative study. In: International Symposium on Graph Drawing and Network Visualization. pp. 179–192. Springer (2019)
4. Chen, Fati and Piccinini, Laurent and Poncelet, Pascal and Sallaberry, Arnaud: Node Overlap Removal Algorithms: an Extended Comparative Study. Journal of Graph Algorithms and Applications (2020)
5. Dwyer, Tim: Scalable, versatile and simple constrained graph layout. In: Computer graphics forum. vol. 28, pp. 991–998. Wiley Online Library (2009)
6. Dwyer, Tim and Marriott, Kim and Stuckey, Peter J: Fast node overlap removal. In: International Symposium on Graph Drawing. pp. 153–164. Springer (2005)
7. Dwyer, Tim and Marriott, Kim and Stuckey, Peter J: Fast node overlap removal—correction. In: International Symposium on Graph Drawing. pp. 446–447. Springer (2006)
8. Gansner, Emden and Hu, Yifan: Efficient, proximity-preserving node overlap removal. Journal of Graph Algorithms and Applications **14**(1), 53–74 (2010)
9. Giovannangeli, Loann and Lalanne, Frederic and Auber, David and Giot, Romain and Bourqui, Romain: Deep Neural Network for DrawiNg Networks, (DNN)^2. In: International Symposium on Graph Drawing and Network Visualization. pp. 375–390. Springer (2021)

10. Hachul, Stefan and Jünger, Michael: Drawing large graphs with a potential-field-based multilevel algorithm. In: International Symposium on Graph Drawing. pp. 285–295. Springer (2004)

11. Harel, David and Koren, Yehuda: Drawing graphs with non-uniform vertices. In: Proceedings of the Working Conference on Advanced Visual Interfaces. pp. 157–166 (2002)

12. Hayashi, Kunihiko and Inoue, Michiko and Masuzawa, Toshimitsu and Fujiwara, Hideo: A layout adjustment problem for disjoint rectangles preserving orthogonal order. In: International Symposium on Graph Drawing. pp. 183–197. Springer (1998)

13. Hu, Yifan: Efficient, high-quality force-directed graph drawing. Mathematica journal **10**(1), 37–71 (2005)

14. Huang, Xiaodi and Lai, Wei and Sajeev, ASM and Gao, Junbin: A new algorithm for removing node overlapping in graph visualization. Information Sciences **177**(14), 2821–2844 (2007)

15. Jakobsen, Thomas: Advanced character physics. In: Game developers conference. vol. 3, pp. 383–401. IO Interactive, Copenhagen Denmark (2001)

16. Kamada, Tomihisa and Kawai, Satoru and others: An algorithm for drawing general undirected graphs. Information processing letters **31**(1), 7–15 (1989)

17. Kamps, Thomas and Kleinz, Joerg and Read, John: Constraint-based spring-model algorithm for graph layout. In: International Symposium on Graph Drawing. pp. 349–360. Springer (1995)

18. Meulemans, Wouter: Efficient optimal overlap removal: Algorithms and experiments. In: Computer Graphics Forum. vol. 38, pp. 713–723. Wiley Online Library (2019)

19. Misue, Kazuo and Eades, Peter and Lai, Wei and Sugiyama, Kozo: Layout adjustment and the mental map. Journal of Visual Languages & Computing **6**(2), 183–210 (1995)

20. Nachmanson, Lev and Nocaj, Arlind and Bereg, Sergey and Zhang, Leishi and Holroyd, Alexander: Node overlap removal by growing a tree. In: International Symposium on Graph Drawing and Network Visualization. pp. 33–43. Springer (2016)

21. Ortmann, Mark and Klimenta, Mirza and Brandes, Ulrik: A sparse stress model. In: International Symposium on Graph Drawing and Network Visualization. pp. 18–32. Springer (2016)

22. Strobelt, Hendrik and Spicker, Marc and Stoffel, Andreas and Keim, Daniel and Deussen, Oliver: Rolled-out wordles: A heuristic method for overlap removal of 2d data representatives. In: Computer Graphics Forum. vol. 31, pp. 1135–1144. Wiley Online Library (2012)

23. Tipping, Michael E and Bishop, Christopher M: Probabilistic principal component analysis. Journal of the Royal Statistical Society: Series B (Statistical Methodology) **61**(3), 611–622 (1999)

24. Van der Maaten, Laurens and Hinton, Geoffrey: Visualizing data using t-SNE. Journal of machine learning research **9**(11) (2008)

25. Wang, Xiaoqi and Yen, Kevin and Hu, Yifan and Shen, Han-Wei: DeepGD: A Deep Learning Framework for Graph Drawing Using GNN. IEEE Computer Graphics and Applications **41**(5), 32–44 (2021)

26. Zheng, Jonathan X and Pawar, Samraat and Goodman, Dan FM: Graph drawing by stochastic gradient descent. IEEE transactions on visualization and computer graphics **25**(9), 2738–2748 (2018)