Journal of Graph Algorithms and Applications
http://jgaa.info/ vol. 24, no. 4, pp. 683–706 (2020)
DOI: 10.7155/jgaa.00532

Node Overlap Removal Algorithms: an Extended Comparative Study

Fati Chen[1,2] *Laurent Piccinini*[2] *Pascal Poncelet*[1]
Arnaud Sallaberry[1,2]

[1]LIRMM, Université de Montpellier, CNRS, Montpellier, France
[2]Groupe AMIS, Université Paul-Valéry Montpellier 3, Montpellier, France

Abstract

In the context of graph layout, many algorithms have been designed to remove node overlapping, and many quality criteria and associated metrics have been proposed to evaluate those algorithms. Unfortunately, a complete comparison of the algorithms based on some metrics that evaluate their quality has never been provided and it is thus difficult for a visualisation designer to select the algorithm that best suits their needs. In this paper, we review 22 metrics available in the literature, classify them according to the quality criteria they try to capture, and select a representative one for each class. Based on the selected metrics,we compare 9 node overlap removal algorithms. Our experiment involves 854 synthetic and real-world graphs. Finally, we propose a JavaScript library containing both the algorithms and the criteria, and we provide a Web platform, *AGORA*, in which one can upload graphs, apply the algorithms and compare/download the results.

Submitted:	Reviewed:	Revised:	Accepted:
October 2019	January 2020	April 2020	May 2020
	Final:	Published:	
	May 2020	December 2020	

Article type: Regular paper
Communicated by: D. Archambault and C. Tóth

Research supported by national French grant ANR Daphne 17-CE28-0013-01

E-mail addresses: fati.chen@lirmm.fr (Fati Chen) laurent.piccinini@univ-montp3.fr (Laurent Piccinini) pascal.poncelet@lirmm.fr (Pascal Poncelet) arnaud.sallaberry@lirmm.fr (Arnaud Sallaberry)

1 Introduction

Graph-drawing algorithms are good at creating rich expressive graph layouts but often consider nodes as points with no dimensions. After changing the size of nodes in the case of annotation or evolving weighted graphs, it causes node overlap which hides information. Post-processing algorithms, named *layout adjustment* [21], have been proposed to remove node overlap.

The objective of these algorithms is, given an initial positioning of the nodes and a size for each one, to provide a new embedding so that there are no overlapping nodes any more. A classical zoom-in function maintaining the sizes of the nodes (i.e., uniform scaling) provides such an embedding, but it expands the visualisation, resulting in large areas without any objects. Therefore, a node overlap removal algorithm must take into account the area of the drawing, and try to minimise it. Positioning the nodes evenly on a grid meets this objective but will result in the loss of the user's mental picture[1] of the original embedding. Thus, it is also important to minimise the change on the layout.

Since a preliminary work in 1995 [21], many algorithms have been designed to reach these goals, and many quality criteria have been proposed to evaluate them. Unfortunately, a complete comparison of the algorithms based on the different criteria has never been provided and it is thus difficult for a visualisation designer to select the one that best suits his needs.

In this paper, our contribution comes in three forms: (1) We propose a classification of 22 quality metrics, grouping them according to the quality criterion they try to capture. We also discuss their relevance and we select a representative one for each class. (2) We compare state-of-the-art node overlap removal approaches in regards to the previously selected metrics. Experiments involve 854 graphs, including synthetic ones (random, tree, scale-free, small-world) and real-world ones. (3) We present a *JavaScript* library[2], that contains all the algorithms described in this paper, and a Web platform, AGORA[3] (*Automatic Graph Overlap Removal Algorithms*), in which one can upload a set of graphs, apply the node overlap removal algorithms and download the results and the values of the quality criteria[4].

The paper is organised as follows: after a brief reminder in Section 2 of the definitions and the notations used in this paper, we present and discuss the quality criteria and the metrics in Section 3. Then we compare the algorithms in Section 4. We discuss threats of validity and future research directions in Section 5. Finally we describe the Web platform in Section 6 and we conclude in Section 7.

[1]An interesting discussion about the concept of mental map preservation is available in [1].

[2]`https://github.com/agorajs/agorajs.github.io` (accessed: 2020-03)

[3]`https://agorajs.github.io/` (accessed: 2020-03)

[4]A preliminary version of this work has been published in the proceedings of the Symposium on Graph Drawing and Network Visualisation 2019 [3]. This new version includes further details on some criteria, a new node overlap removal algorithm (*Diamond* [20]), a more detailed analysis of the results, a discussion on the threats to validity and the directions for future work, the library and the Web platform.

2 Preliminaries

In this paper, we use the following definitions and notations.

$G = (V, E)$ denotes a graph where V is a set of nodes and E a set of edges. The number of nodes $|V|$ is denoted by n and the number of edges $|E|$ by m. We consider each node as a rectangle. Thus, for a node $v \in V$, its width and its height are denoted by the couple (w_v, h_v) which is not changed by the layout adjustment.

The initial embedding is defined as an injection $\mathcal{E}_G$: $V \to \mathbb{R}^2$ such that $\forall v \in V$, $\mathcal{E}_G(v) = (x_v, y_v)$ where (x_v, y_v) are the coordinates of the center of the node v. The overlapping-free embedding is denoted by $\mathcal{E}'_G$. To simplify notations, we denote $v = (x_v, y_v)$ instead of $\mathcal{E}_G(v)$, and $v' = (x'_v, y'_v)$ instead of $\mathcal{E}'_G(v)$. Remark that two nodes $(u, v) \in V^2$ are overlapping when:

$$|x_v - x_u| < \frac{w_v + w_u}{2} \quad \text{and} \quad |y_v - y_u| < \frac{h_v + h_u}{2}$$

The bounding box bb of an embedding $\mathcal{E}_G$ is defined as the smallest rectangle containing all the nodes of G; w_{bb} (resp. h_{bb}) denotes the width (resp. the height) of the initial embedding, w'_{bb} (resp. h'_{bb}) denotes the width (resp. the height) of the overlapping-free one. They are determined as follows:

$$w_{bb} = \left| \max_{v \in V} \left(x_v + \frac{w_v}{2} \right) - \min_{u \in V} \left(x_u - \frac{w_u}{2} \right) \right| \tag{1}$$

$$h_{bb} = \left| \max_{v \in V} \left(y_v + \frac{h_v}{2} \right) - \min_{u \in V} \left(y_u - \frac{h_u}{2} \right) \right| \tag{2}$$

The position of the center of the bounding box is denoted by $c_{bb} = (x_{bb}, y_{bb})$ in the initial embedding, and $c'_{bb} = (x'_{bb}, y'_{bb})$ in the overlapping-free embedding.

The convex hull of an embedding $\mathcal{E}_G$ is defined as the smallest convex region containing all the nodes of G. Note that it is computed by using the 4 corners of the nodes, and not only their center, in a way that the rectangles representing the nodes are fully included into it. In the following, ch denotes the convex hull of the original embedding, ch' the convex hull of the free-overlapping one, c_{ch} the center of mass of ch, c'_{ch} the center of mass of ch'.

3 Quality criteria

Many criteria have been proposed in the literature to evaluate the quality of the embeddings resulting from adjustment algorithms. Unfortunately, the experiments provided by the authors of the different approaches are not always based on the same metrics. In order to provide a uniform protocol of experiment and a complete comparison of the algorithms, we need to review the quality criteria and the metrics used to evaluate them. We also need to select a representative metric for each criterion.

We identified 5 classes of metrics: *Orthogonal Ordering preservation* (*oo*), *Spread minimisation* (*sp*), *Global Shape preservation* (*gs*), *Node Movement minimisation* (*nm*) and *Edge Length preservation* (*el*). Each of them depicts a quality criterion. Table 1 shows the metrics assigned to the classes. The formulas are given in the discussion below. The abbreviations of the classes are used as prefix for the metrics.

The following subsections contain the metrics of a specific class. In each of them, we select one representative metric, based on the corresponding quality criterion and the properties that the metrics aim at capturing. Our discussion also sometimes involves the coefficient of correlation of two metrics run following the protocol described in the comparison section, Section 4.

3.1 Orthogonal Ordering preservation

The orthogonal ordering class groups the metrics which try to quantify how much an adjustment algorithm preserves the initial orthogonal ordering, i.e., the following conditions:

$$\begin{cases} x_u < x_v \Leftrightarrow x'_u < x'_v \\ y_u < y_v \Leftrightarrow y'_u < y'_v \\ x_u = x_v \Leftrightarrow x'_u = x'_v \\ y_u = y_v \Leftrightarrow y'_u = y'_v \end{cases}$$

The first metric of this class introduced in [21], called here *oo_o*, is equal to 1 if the overlapping-free graph embedding preserves the initial orthogonal ordering, 0 otherwise. Also, if only one couple of nodes does not satisfy those conditions, the value of *oo_o* is the same as when many ones do not satisfy it.

To overcome this issue, Huang *et al.* [16] proposed a metric based on the *Kendall's Tau distance* (*oo_kt*). For each couple of nodes, they first compute an inversion number $inv(u, v)$ corresponding to 0 if the orthogonal ordering is preserved between them, 1 otherwise. The metric is then defined as the normalised sum of the inversion numbers:

$$\text{oo\_kt} = \frac{\sum\limits_{u \neq v} inv(u, v)}{n(n-1)}$$

Strobelt *et al.* [26] introduced the number of inversions:

$$\text{oo\_ni} = \sum_{\substack{(u,v)\in V^2 \\ x_u > x_v}} \begin{cases} 1 & \text{if } x'_u < x'_v \\ 0 & \text{otherwise} \end{cases} + \sum_{\substack{(u,v)\in V^2 \\ y_u > y_v}} \begin{cases} 1 & \text{if } y'_u < y'_v \\ 0 & \text{otherwise} \end{cases}$$

Table 1: List of metrics classified by the quality criterion they try to capture: selected metrics appear in bold italics. The abbreviations are based on some initials of the names, *e.g.* *sp_bb_a* means that the metric is in the class *Spread minimisation*, it uses the embedding *Bounding Box* to quantify the *Area* spreading. The *Range* column contains the set of values that the metric can take. The *Tgt* column refers to the target value to meet the corresponding criterion.

Abbrev.	**Name**	**Range**	**Tgt**
	Orthogonal Ordering preservation		
oo_o	Original [21]	$\{0, 1\}$	1
oo_kt	Kendall's Tau Distance [16]	$[0, 1]$	0
oo_ni	Number of Inversions [26]	$[0, n(n-1)]$	0
oo_nni	***Normalised Number of Inversions***	$[0, 1]$	0
	Spread minimisation		
sp_bb_l1ml	Bounding Box L1 Metric Length [17]	$[1, +\infty[$	1
sp_bb_a	Bounding Box Area [21]	$[1, +\infty[$	1
sp_bb_na	Bounding Box Normalised Area [16]	$[0, 1[$	0
sp_ch_a	***Convex Hull Area*** [26]	$[1, +\infty[$	1
	Global Shape preservation		
gs_bb_ar	Bounding Box Aspect Ratio [17]	$]0, +\infty[$	1
gs_bb_iar	***Bounding Box Improved Aspect Ratio***	$[1, +\infty[$	1
gs_ch_sd	Convex Hull Standard Deviation [26]	$[0, +\infty[$	0
	Node Movement minimisation		
nm_mn	Moved Nodes [16]	$[0, 1]$	0
nm_dm_me	Distance Moved Mean Euclidean [26]	$[0, +\infty[$	0
nm_dm_ne	Distance Moved normalised Euclidean [18]	$[0, 1]$	0
nm_dm_h	Distance Moved Hamiltonian [15, 16]	$[0, +\infty[$	0
nm_dm_se	Distance Moved Squared Euclidean [19]	$[0, +\infty[$	0
nm_dm_imse	***Distance Moved Improved Mean Squared Euclidean***	$[0, +\infty]$	0
nm_d	Displacement [8]	$]0, +\infty[$	0
nm_knn	K-Nearest Neighbours [22]	$[0, +\infty[$	0
	Edge Length preservation		
el_r	Ratio [17]	$[1, +\infty[$	1
el_rsdd	*Relative Standard Deviation Delaunay* [8]	$[0, +\infty]$	0
el_rsd	***Relative Standard Deviation***	$[0, +\infty]$	0

This metric has the drawback of providing non-normalised values. However, it holds the benefit of penalizing inversions occurring on each axis independently ($x-$ and $y-axis$), instead of penalizing in the same manner an inversion occurring in only one axis and an inversion occurring in the two axes. Thus, in our study, we combine the two metrics by using a normalised version of the latter:

$$\text{oo\_nni} = \frac{oo\_ni}{n(n-1)}$$

3.2 Spread minimisation

A classical zoom-in function maintaining the sizes of the nodes (i.e. uniform scaling) provides an overlapping-free embedding, but it expands the visualisation, resulting in large areas without any objects. To avoid this issue, quality metrics have been introduced to quantify embedding spreading. Their purpose is to favour algorithms inducing low spreading.

The L1 metric length [17] is the ratio:

$$\text{sp\_bb\_l1ml} = \frac{\max(w'_{bb}, h'_{bb})}{\max(w_{bb}, h_{bb})}$$

The drawback of this technique is to consider only one dimension of the embedding, width or height. For instance, considering an example where $w_{bb} = 4$, $h_{bb} = 2$, $w'_{bb} = 4$, $h'_{bb} = 4$, the value of the L1 metric length is 1 (which is the target value), whereas the area of the overlapping-free embedding is twice as large as in the initial embedding. The ratio between the bounding box areas of the two embeddings [21] overcomes this issue:

$$\text{sp\_bb\_a} = \frac{w'_{bb} \times h'_{bb}}{w_{bb} \times h_{bb}}$$

While the result gives an unbounded value greater than 1, Huang *et al.* [16] proposes a normalised version producing values in the interval $[0, 1[$:

$$\text{sp\_bb\_na} = 1 - \frac{w_{bb} \times h_{bb}}{w'_{bb} \times h'_{bb}}$$

Unfortunately, this criterion is rather unintuive and it is hard to figure out what the values represent.

In our comparison, we selected another version of the ratio of areas involving convex hulls [26], as it better captures the concrete area of the drawing:

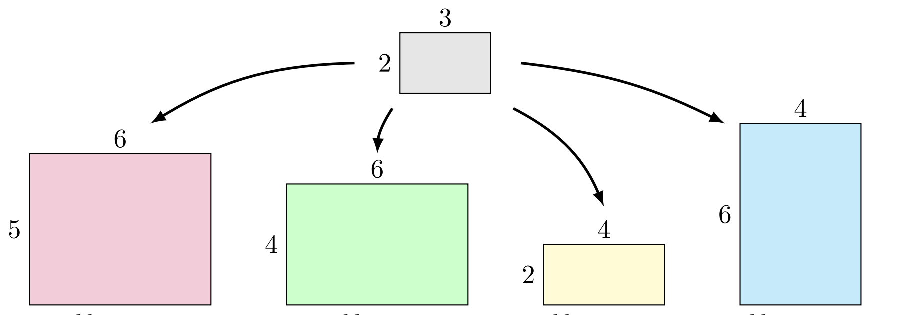
$$\text{sp\_ch\_a} = \frac{area(ch')}{area(ch)}$$

3.3 Global Shape preservation

This class contains metrics that try to capture the ability of the algorithms to preserve the global shape of the initial embedding. The first one was proposed by Li *et al.* [17]:

$$\text{gs\_bb\_ar} = \begin{cases} \dfrac{w'_{bb} \times h_{bb}}{h'_{bb} \times w_{bb}} & \text{if } w'_{bb} > h'_{bb} \\ \dfrac{h'_{bb} \times w_{bb}}{w'_{bb} \times h_{bb}} & \text{otherwise} \end{cases}$$

The underlying idea is to capture the variation of the aspect ratio (w_{bb}/h_{bb}) between the initial and the overlapping-free embedding. For instance, let us consider an example where $w_{bb} = 3$, $h_{bb} = 2$, $w'_{bb} = 6$, $h'_{bb} = 4$ (see grey and green rectangles below). In this case, the overlapping-free embedding is twice as large as the initial one but the aspect ratio remains the same 3/2. The *gs_bb_ar* is 1, which is the target value. Now let us consider another example where $w_{bb} = 3$, $h_{bb} = 2$, $w'_{bb} = 4$, $h'_{bb} = 6$ (blue rectangle below). In this case, the initial aspect ratio is 3/2 whereas the overlapping-free one is 2/3. The *gs_bb_ar* is now 2.25, which is not the target value; it reveals a distortion of the initial embedding during the overlap removal process.



$gs\_bb\_ar = 0.8$ $gs\_bb\_iar = 1.25$ | $gs\_bb\_ar = 1$ $gs\_bb\_iar = 1$ | $gs\_bb\_ar = 1.33$ $gs\_bb\_iar = 1.33$ | $gs\_bb\_ar = 2.25$ $gs\_bb\_iar = 2.25$

The main drawback of this metric is that it can reach values in the interval $]0, +\infty[$ while the target value is 1. For instance, $w_{bb} = 3$, $h_{bb} = 2$, $w'_{bb} = 6$ and $h'_{bb} = 5$ induce a *gs_bb_ar* equals to 0.8 (purple rectangle above), while $w_{bb} = 3$, $h_{bb} = 2$, $w'_{bb} = 4$ and $h'_{bb} = 2$ induce a *gs_bb_ar* equals to 1.33 (yellow rectangle above). In this case, it is hard to decide which algorithm is the best between two of them if the first one obtains the purple bounding box and the second one obtains the yellow one, because we have no clue to compare 0.8 and 1.33 when 1 is the target value. To overcome this issue, we propose to refine it as follows:

$$\text{gs\_bb\_iar} = \max\left(\frac{w'_{bb} \times h_{bb}}{h'_{bb} \times w_{bb}}, \frac{h'_{bb} \times w_{bb}}{w'_{bb} \times h_{bb}}\right)$$

In this case, the target value is 1 and the metric cannot reach values below it (see the values below the rectangles). This criterion is the one we selected for our study.

An alternative to this approach based on the convex hull has been proposed by Strobelt *et al.* [26]. The idea is to evaluate the distortion of the convex hull by comparing, between both embeddings, the distances of convex hull points to their center. Let ℓ_θ (resp. ℓ'_θ) be the Euclidean distance between the center of mass c_{ch} (resp. c'_{ch}) of the convex hull ch (resp. ch') and the intersection of the convex hull with the line going through c_{ch} (resp. c'_{ch}) and with an angle θ (θ varying from 0° to 350° in 10° steps). Then, the difference is defined as the ratio $d_\theta = \ell'_\theta / \ell_\theta$. The metric is the standard deviation of the 36 measures of d_θ:

$$\text{gs\_ch\_sd} = \sqrt{\frac{1}{36} \sum_{\substack{\theta=10k \\ k=0,\cdots,35}} (d_\theta - \overline{d})^2}$$

$$\text{where } \overline{d} = \frac{1}{36} \sum_{\substack{\theta=10k \\ k=0,\cdots,35}} d_\theta \text{ is the mean value}$$

Based on the experiments presented below in Section 4, we observed that *gs_bb_iar* and *gs_ch_sd* have a correlation coefficient of 0.77, showing that they both tend to capture similar aspects of the adjustment process. We selected the former for its simplicity and its ease of interpretation.

3.4 Node Movement minimisation

This class contains the metrics quantifying the changes in node positions after running an adjustment algorithm. The underlying intuition is that an algorithm involving high node movements will provide an overlapping-free configuration different from the original one, and thus may result in a substantial loss of the mental model.

The simplest metric of this class was presented by Huang *et al.* [16]:

$$\text{nm\_mn} = \frac{nb}{n}$$

Here, nb represents the number of nodes which have moved between the initial and the overlapping-free embedding. The main drawback of this approach is that a node overlap removal algorithm may induce very small changes in most nodes, which does not affect the mental model preservation, while inducing a very bad result. To tackle this problem and add more granularity over the evaluation of node movements, a series of metrics have been proposed, based on the same underlying quality function:

$$\text{nm\_dm} = f(n) \times \sum_{v \in V} \text{dist}(v, v')$$

where f is a normalising function of $n = |V|$ and *dist* is a distance between v and v'. Table 2 sums up the ones used in the literature.

The function f comes in three different forms. Marriott *et al.* [19] and Huang *et al.* [15] do not include any f, which is similar to having $f(n) = 1$. The drawback is that the resulting value highly depends on the number of nodes in the graph. That is why Strobelt *et al.* [26] proposed to use the mean of

Table 2: Functions used to tune the distance moved metric (with references to the first papers mentioning the use of these functions in the context of node overlap removal)

$\text{dist}(v,v')\ \ \backslash\ \ f(n)$	1	$1/n$	$\frac{1}{k\sqrt{2}\times n}$
$\lVert v'-v\rVert$		*nm_dm_me* [26]	*nm_dm_ne* [18]
$\lVert v'-v\rVert^2$	*nm_dm_se* [19]	*nm_dm_imse*	
$\lvert x'_v - x_v\rvert + \lvert y'_v - y_v\rvert$	*nm_dm_h* [15]		

.

the distances, which corresponds to $f(n) = 1/n$. Finally, Lyons *et al.* [18] proposed $f(n) = 1/(k\sqrt{2} \times n)$, where k is the maximum between w'_{bb} and h'_{bb}. In this case, $k\sqrt{2}$ is the diagonal of a square containing the embedding, thus a maximum distance available for a node. Unfortunately, this normalisation generates very small values and is harder to interpret than $f(n) = 1/n$. That is why we preferred the latter for our study.

Three *dist* functions have been proposed in the literature. The most intuitive one is the Euclidean distance $\lVert v' - v\rVert$ [26, 18]. The squared Euclidean distance $\lVert v' - v\rVert^2$ [19] avoids the square root computation and discriminates high changes better. It is the one we selected for our study. The Manhattan distance $\lvert x'_v - x_v\rvert + \lvert y'_v - y_v\rvert$ has also been used [15], but it is less intuitive and has close results (*nm_dm_se* and *nm_dm_h* have a correlation coefficient of 0.9).

Let us consider an adjustment algorithm that pushes nodes on the x-axis. The preservation of the global shape is not optimal but the preservation of the configuration should reach a good score, as a node on right-top in the initial embedding would remain on right-top in the overlapping-free embedding. In order to better capture the relative movement of a node between the two embeddings, a *shift* function can be applied to align the center of the initial bounding box with the center of the final one, and a *scale* function to align the size of the initial bounding box to the size of the final one:

$$\begin{aligned} shift(v) &= (x_v + x'_{bb} - x_{bb}, y_v + y'_{bb} - y_{bb}) \\ scale(v) &= (x_v \times \frac{w'_{bb}}{w_{bb}}, y_v \times \frac{h'_{bb}}{h_{bb}}) \end{aligned}$$

Considering this, we selected the following node movement metric:

$$\text{nm\_dm\_imse} = \frac{1}{n} \times \sum_{v \in V} \lVert v' - scale(shift(v))\rVert^2$$

nm_d [8] (the complete formula is available in the paper) is also based on the idea that the metric should be based on modified initial positions to better capture the relative movement of the nodes between the two embeddings. Besides including the *shift* and the *scale* functions, it also rotates the initial embedding with an angle θ that minimises the distances between the nodes of the initial embedding and the ones of the overlapping-free embedding:

$$rotation(v) = (x_v \cos\theta - y_v \sin\theta, x_v \sin\theta + y_v \cos\theta)$$

We have not included the rotation in our experiment as we consider that it can induce a loss of the mental model (think about the recognition of a map turned upside down).

An alternative to quantify how much an overlapping-free configuration may result in a substantial loss of the mental model is to look at the neighbourhoods at the nodes and compare them before and after the adjustment. Based on a k-NN approach, Nachmanson *et al.* [22] proposed the following metric:

$$nm\_knn(k) = \sum_{v \in V} \left(k - |N_k(v) \cap N_k(v')|\right)^2$$

where $N_k(v)$ (resp. $N_k(v')$) denotes the k nearest neighbours of v (resp. v'), in terms of Euclidean distance, in the initial (resp. overlapping-free) embedding. We did not select this metric because, unlike the other metrics of the class, it requires to fix a parameter (k).

3.5 Edge Length preservation

This class contains the two metrics based on edge lengths. The set of edges can be E or can be another set derived from the graph.

Standard force-based layout algorithms tend to produce uniform lengths of edges. Indeed, the first metric of this class captures whether the edge lengths of a graph remain uniform or not after applying an adjustment algorithm [17]:

$$el\_r = \frac{\max_{(u,v)\in E} \|u' - v'\|}{\min_{(u,v)\in E} \|u' - v'\|}$$

As many layout algorithms are not designed to produce uniform edge lengths, mental map preservation is not necessarily captured by such kind of metrics. Hence we decided to consider alternatives.

The first alternative is based on the edges of a graph derived from the initial embedding, via a Delaunay triangulation. Let E_{dt} be the set of edges of a Delaunay triangulation performed on the nodes of the initial embedding. The second metric of this class, *el_rsdd*, is based on computing the coefficient of variation, also known as the relative standard deviation, of the edge lengths ratio as follows [8]:

$$r_{uv} = \frac{||u' - v'||}{||u - v||}, \quad (u, v) \in E_{dt}$$

$$\overline{r} = \frac{1}{|E_{dt}|} \sum_{(u,v)\in E_{dt}} r_{uv}$$

$$el\_rsdd = \frac{\sqrt{\frac{1}{|E_{dt}|} \sum_{(u,v)\in E_{dt}} (r_{uv} - \overline{r})^2}}{\overline{r}}$$

The main drawback of this metric is that it is based on a derived set of edges, instead of the real one. As a consequence, it only partially captures whether an algorithm preserves edge lengths or not. In our study, we use the coefficient of variation of edge lengths ratio, *el_rsd* (same equations as *el_rsdd* replacing E_{dt} by E).

4 Algorithms comparison

In this section, we compare 9 algorithms of the literature in terms of quality and running time: uniform *Scaling*, *PFS* [21], *PFS'* [13], *FTA* [16], *VPSC* [5], *PRISM* [8], *RWordle-L* [26], *GTREE* [22], and *Diamond* [20]. The quality of an overlapping-free embedding is evaluated with the metrics identified in the last section, by following a 3 steps procedure:

1. **Datasets.** We generate 840 synthetic graphs containing 10 to 1,000 nodes. These graphs are provided by 4 generation models available on the OGDF library [4]: random graphs [6], random trees, small world graphs [28], and scale-free graphs [2]. We also use 14 real-world graphs selected from the Graphviz test suite[5] [9], previously used by the authors of *PRISM* [8] and *GTREE* [22]. All the graphs are available online[6] as *GML* files including the initial embedding.
2. **Overlapping-free embedding computation.** Synthetic graphs resulting from the first step are initially positioned by the FM^3 layout algorithm [11]. Then, we apply the 9 node overlap removal algorithms, thus providing a set of 7.560 overlapping-free graph embeddings. `Graphviz` test suite graphs are initially positioned by the $SFDP$ layout algorithm [14] to follow the same baseline embedding as Gansner *et al.* [8]. We then apply the 9 node overlap removal algorithms thus providing 126 overlapping-free graph embeddings.
3. **Metrics computation.** We finally compute the values of the 5 selected metrics on the 7.686 overlapping-free synthetic and real-world graph embeddings. We also measure the computation time of the algorithms.

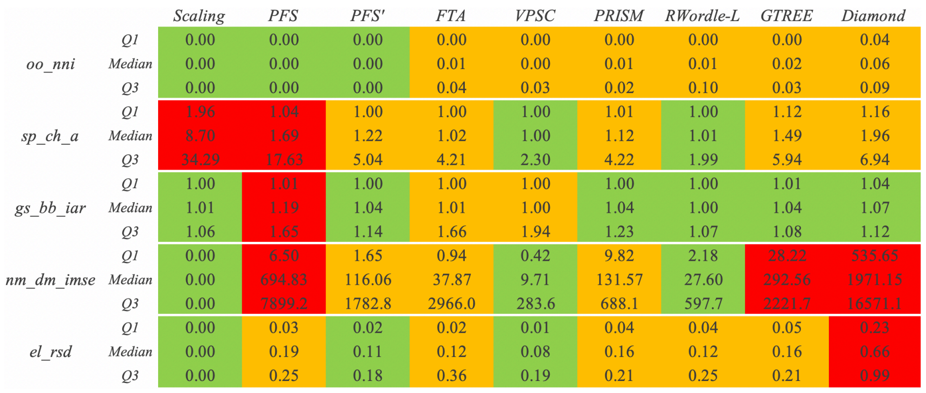
The values of the metrics discussed in this comparison are measured from the results of the implemented algorithms that are available in our library (see Section 6), and thus might differ from our original paper [3] in terms of running time. All the algorithms are coded in *JavaScript*. We implemented *PFS*, *PFS'*, *FTA* and *Diamond* ourselves from the algorithms provided by the authors in their seminal papers. As *Diamond* is based on a linear optimization, we used the *jsLPSolver*[7]. For *VPSC*, we directly used the *JavaScript* program provided by the authors[8]. *PRISM* and *GTREE* have been adapted from the *Microsoft*

[5]`https://gitlab.com/graphviz/graphviz/blob/master/rtest/graphs/` (accessed: 2020-03)

[6]`https://github.com/agorajs/agora-dataset`(accessed: 2020-03)

[7]`https://github.com/JWally/jsLPSolver` (accessed: 2020-03)

[8]`https://github.com/tgdwyer/WebCola` (accessed: 2020-03)



		Scaling	*PFS*	*PFS'*	*FTA*	*VPSC*	*PRISM*	*RWordle-L*	*GTREE*	*Diamond*
oo_nni	*Q1*	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.04
	Median	0.00	0.00	0.00	0.01	0.00	0.01	0.01	0.02	0.06
	Q3	0.00	0.00	0.00	0.04	0.03	0.02	0.10	0.03	0.09
sp_ch_a	*Q1*	1.96	1.04	1.00	1.00	1.00	1.01	1.00	1.12	1.16
	Median	8.70	1.69	1.22	1.02	1.00	1.12	1.01	1.49	1.96
	Q3	34.29	17.63	5.04	4.21	2.30	4.22	1.99	5.94	6.94
gs_bb_iar	*Q1*	1.00	1.01	1.00	1.00	1.00	1.00	1.00	1.01	1.04
	Median	1.01	1.19	1.04	1.01	1.00	1.04	1.00	1.04	1.07
	Q3	1.06	1.65	1.14	1.66	1.94	1.23	1.07	1.08	1.12
nm_dm_imse	*Q1*	0.00	6.50	1.65	0.94	0.42	9.82	2.18	28.22	535.65
	Median	0.00	694.83	116.06	37.87	9.71	131.57	27.60	292.56	1971.15
	Q3	0.00	7899.2	1782.8	2966.0	283.6	688.1	597.7	2221.7	16571.1
el_rsd	*Q1*	0.00	0.03	0.02	0.02	0.01	0.04	0.04	0.05	0.23
	Median	0.00	0.19	0.11	0.12	0.08	0.16	0.12	0.16	0.66
	Q3	0.00	0.25	0.18	0.36	0.19	0.21	0.25	0.21	0.99

Table 3: Aggregated values of the selected metrics on the synthetic graphs: first quartile, median and third quartile.

Automatic Graph Layout library[9] and converted into *JavaScript* with *SharpKit*[10]. Finally, we were inspired by the *Java* program of *RWordle-L*[11] provided by the authors of [26].

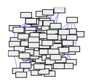
4.1 Quality

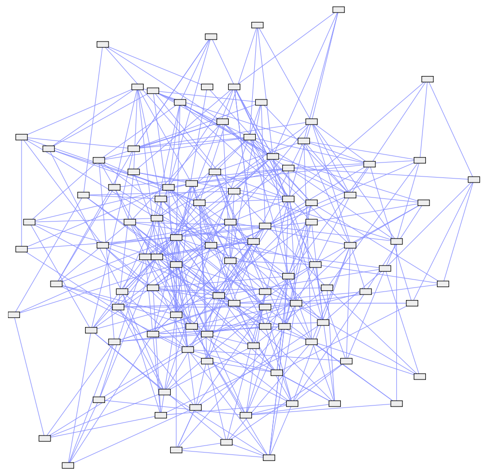
Figure 1a shows a random graph containing 100 nodes and 400 edges, positioned by the FM^3 layout algorithm [11]. This initial embedding contains 274 overlaps. Figures 1b-1j show the overlapping-free embeddings obtained after applying the algorithms mentioned above. The sizes of the figures reflect the spread of the embeddings.

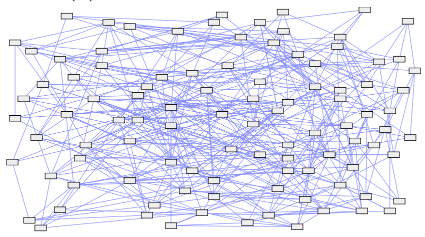
Figure 2 shows another example with a real-world graphs from the Graphviz test suite, *mode*. It contains 213 nodes, 269 edges and 1105 overlaps. Figures 2b-2j shows the overlapping-free embeddings. In this case, we did not maintain the relative size for *Scaling* and *PFS*, as the drawing was too large. The actual embeddings are twice as large as they appear in the figure.

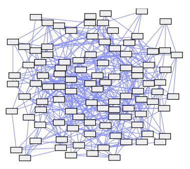
Table 3 shows the aggregated metrics values obtained on the synthetic graphs: for each of the five selected metrics and for each algorithm, the first quartile, the median and the third quartile of the values are given. Table 4 shows the metric values on the real-world graphs. In these figures and the next ones, the colour of the cases represents the quality of the algorithm on the criterion: green for high quality, orange for intermediate and red for poor. The ranges are defined by comparing the values lying on a single row, i.e. the values of a single criterion obtained on the different algorithms.

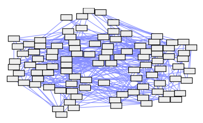
[9] `https://github.com/microsoft/automatic-graph-layout` (accessed: 2020-03)
[10] `https://github.com/SharpKit` (accessed: 2020-03)
[11] `https://github.com/HendrikStrobelt/ditop_server/blob/master/src/main/java/de/hs8/graphics/RWordle.java` (accessed: 2020-03)

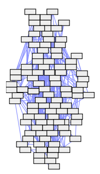

(a) *Initial*

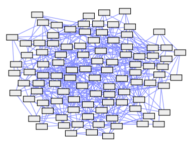

(b) *Scaling*

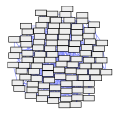

(c) *PFS*

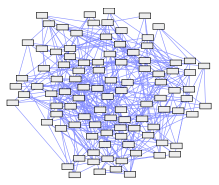

(d) *PFS'*


(e) *FTA*


(f) *VPSC*


(g) *PRISM*


(h) *RWordle-L*


(i) *GTREE*

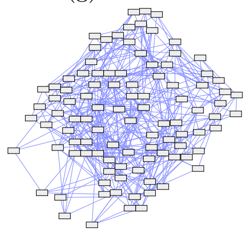
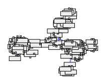
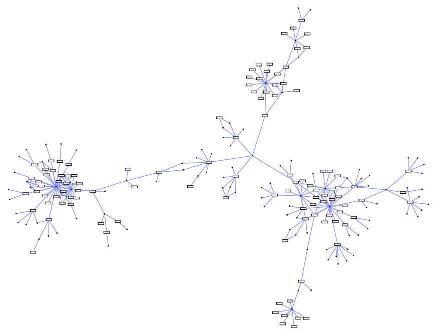
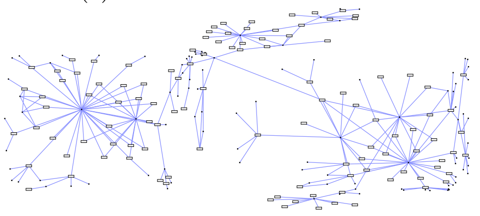
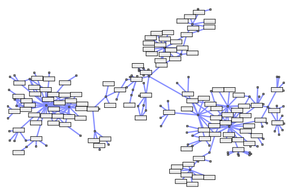

(j) *Diamond*

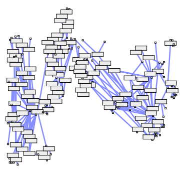
Figure 1: Overlapping-free embeddings obtained after applying the algorithms on the initial embedding (a) of a random graph containing 274 overlaps.

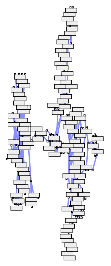

(a) *Initial*

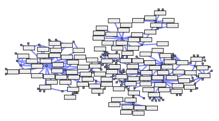

(b) *Scaling*

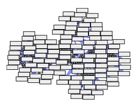

(c) *PFS*

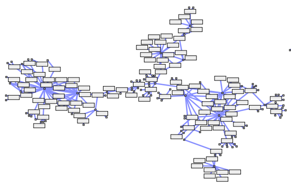

(d) *PFS'*


(e) *FTA*


(f) *VPSC*


(g) *PRISM*


(h) *RWordle-L*


(i) *GTREE*

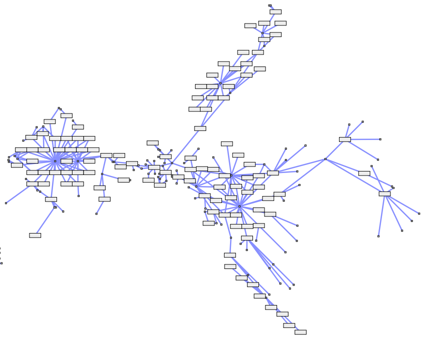

(j) *Diamond*

Figure 2: Overlapping-free embeddings obtained after applying the algorithms on the initial embedding (a) of a real-world graph containing 1105 overlaps.

	Scaling	*PFS*	*PFS'*	*FTA*	*VPSC*	*PRISM*	*RWordle-L*	*GTREE*	*Diamond*
oo_nni	0.00	0.00	0.00	0.07	0.01	0.02	0.04	0.02	0.05
sp_ch_a	217.26	210.28	6.92	6.03	2.36	2.18	1.53	4.02	30.63
gs_bb_iar	1.04	1.97	1.22	1.95	2.20	1.33	1.04	1.17	1.10
nm_dm_imse	0.00	9768157.94	63594.74	4297355.84	34611.40	42919.66	35928.91	37331.83	1348426.00
el_rsd	0.00	0.34	0.22	0.54	0.28	0.28	0.36	0.26	0.23

Table 4: Mean values of the selected metrics on the real-world graphs.

Orthogonal Ordering preservation Unsurprisingly, *Scaling*, *PFS* and *PFS'* obtain the best scores at *oo_nni* as it is proved that they maintain the original orthogonal ordering. Though, all the algorithms tested got good results for this criterion.

Spread minimisation As shown in Figure 1b, *Scaling* highly increases the size of the embedding, which induces a bad score for *sp_ch_a*. *PFS* also obtains a bad score for this criterion. *VPSC* and *RWordle-L* produce the most compact embeddings, while the other algorithms give intermediary results. However, looking at Figures 1f, 1h, 2f and 2h, we can observe that the embeddings resulting from these two algorithms are so compact that they do not allow to visualise the edges nor the structures of the graph (e.g. communities or clusters). Depending on the task one wants to perform on the overlapping-free embedding, this observation illustrates a possible limitation of the criterion when it is considered independently from the other ones.

Global Shape preservation Surprisingly, the global shape preservation score (*gs_bb_iar*) is not exactly 1 for *Scaling* because of the size of the nodes that remains the same between the initial and the overlapping-free embeddings. Nevertheless, it preserves the initial global shape. *PFS* is the worst algorithm on this criterion. The other algorithms obtained good median scores on synthetic graphs, but the third quartile scores show that *FTA* and *VPSC* can produce a certain amount of distorted embeddings. This is confirmed by the tests on real-world graphs, where they obtain worse results, and on the Figures 1e, 1f and 2f, where we can observe that they spread the layout along only one of the axis (x-axis for *FTA* and y-axis for *VPSC*).

Node Movement minimisation *Scaling* obtains the best results for the node movement minimisation criterion, followed by *VPSC* and *RWordle-L*. *FTA* also obtained a good median score on synthetic graphs, but its third quartile value shows that it can generate a certain amount of embeddings with high changes, as also illustrated by the bad score obtained on the real-world graphs. *PFS'* and *PRISM* obtained intermediary results. *GTREE* had bad results on the synthetic graphs, while it obtained pretty good ones on the real-world graphs. Finally, *PFS* and *Diamond* obtained bad results on both synthetic and real-world graphs.

Edge Length preservation *Scaling* preserves relative edge lengths. *Diamond* obtains the worst scores on synthetic graphs but this phenomenon is not confirmed on real-world ones, for which it obtains pretty good scores. All the other algorithms obtained a median score between 0.08 and 0.36 on synthetic graphs. The third quartile shows that *FTA* generates a certain amount of embeddings with higher edge length variations. This observation is confirmed by the results on the real-world graphs, for which it obtains the worst score.

4.2 Computation time

Tables 5 and 6 show the aggregated running time values in milliseconds on the synthetic graphs (first quartile, median and third quartile) and the running time values on the real-world ones, measured on our implementation of the algorithms.

We can observe on the synthetic graphs that *Scaling*, *PFS*, *PFS'* and *VPSC* require lower running time than the other algorithms. *FTA* is a little bit slower, especially when looking at the third quartile, indicating a certain amount of time consuming embedding computations (more than 1 second for graphs containing more than 500 nodes). This observation is confirmed on the real-world graphs.

RWordle-L, *PRISM* and *GTREE* induce intermediate running times on the synthetic graphs: less than 1 second for graphs containing up to 200 nodes, a few seconds for graphs of 500 nodes, and tens of seconds for graphs of 1000 nodes. *PRISM* and *GTREE* are significantly slower than *RWordle-L* on small graphs (number of nodes below or equal to 100) but it seems to be unimportant as the values remain very low. The real-world graphs confirm these observations, but also highlight that *PRISM* is sometimes significantly slower than *GTREE*, even if it only happens on graphs requiring a low time computation.

Diamond is often the most time consuming algorithm on both synthetic and real-world graphs. However, it is based on a linear optimization so the running time depends on the solver used (see introduction of this section). This might explain the differences between our results and the ones given in the paper [20].

4.3 Summary

As a conclusion, even if *Scaling* optimises 4 out of 5 criteria and is very fast to compute on the graphs of our datasets, it does not represent a satisfying solution as it increases the size of the embedding too much. *PFS* is also not satisfying as it got poor results on 3 criteria. In particular, it also considerably increases the size of the embedding, which is obvious in Figures 1c and 2c.

FTA obtained intermediate results over all the criteria, which is less good than all its remaining competitors. In particular, as mentioned before, it spreads the embedding along only one axis, highly distorting the original configuration (see the *Global Shape Preservation* criterion *gs_bb_iar* in Table 3, as well as the distortion illustrated by Figure 1e).

VPSC also holds this property. Looking at Table 3, we can observe that *RWordle-L* is better than *VPSC* on *Global Shape Preservation*, while obtaining

		Scaling	*PFS*	*PFS'*	*FTA*	*VPSC*	*PRISM*	*RWordle-L*	*GTREE*	*Diamond*
	Q1	0.00	0.00	0.00	0.00	0.00	2.00	0.00	3.00	1.00
10	*Median*	0.00	0.00	0.00	0.00	0.00	4.50	0.00	5.00	1.00
	Q3	0.00	0.00	0.00	0.00	0.00	12.00	0.00	12.00	1.00
	Q1	0.00	0.00	0.00	0.00	0.00	7.00	0.00	5.00	3.00
20	*Median*	0.00	0.00	0.00	0.00	0.00	20.00	0.00	12.00	3.00
	Q3	0.00	0.00	0.00	0.00	1.00	28.00	1.00	16.00	4.00
	Q1	0.00	0.00	0.00	0.00	1.00	24.00	0.00	19.00	15.00
50	*Median*	0.00	0.00	0.00	1.00	1.00	56.00	3.00	38.00	22.00
	Q3	0.00	0.00	1.00	2.00	2.00	115.00	9.25	50.00	27.00
	Q1	0.00	0.00	1.00	1.00	2.00	109.50	0.00	85.00	145.80
100	*Median*	0.00	1.00	1.00	4.00	3.00	215.50	36.50	125.00	238.50
	Q3	1.00	1.00	1.00	12.00	4.00	284.00	84.25	156.25	318.50
	Q1	1.00	2.00	4.00	4.00	4.00	431.50	2.00	307.50	3230.00
200	*Median*	1.00	2.00	4.00	25.50	10.50	657.00	274.00	422.00	6358.00
	Q3	2.00	3.00	5.00	89.00	15.00	927.50	589.50	587.50	7960.00
	Q1	2.00	15.00	25.00	26.00	25.00	3007.00	21.00	1930.00	149824.00
500	*Median*	10.00	17.00	29.50	207.50	93.00	4319.00	3410.00	3126.00	302230.00
	Q3	13.00	20.00	35.00	1392.00	140.75	5001.00	7246.25	4060.00	390922.00
	Q1	4.75	57.00	113.50	87.75	125.50	16340.00	112.00	9187.00	3765418.00
1000	*Median*	34.50	67.50	133.00	1022.00	496.00	21108.00	26877.50	16382.00	7450774.00
	Q3	58.25	77.00	176.25	8694.00	1065.75	24908.00	55173.25	19892.00	9775764.00

Table 5: Aggregated running times in milliseconds on the synthetic graphs, function of number of nodes (10 to 1,000): first quartile, median and third quartile.

	Scaling	*PFS*	*PFS'*	*FTA*	*VPSC*	*PRISM*	*RWordle-L*	*GTREE*	*Diamond*
b100	41	82	126	19823	321	138528	191109	32236	44786574
b102	5	16	24	82	15	3196	96	919	52913
b124	0	0	0	3	1	308	1	54	199
b143	0	0	1	9	2	351	6	125	982
badvoro	20	43	63	32553	383	50360	56155	26140	23981749
dpd	0	0	0	1	1	146	0	80	47
mode	0	1	2	93	4	1891	281	422	3960
NaN	0	1	1	2	0	113	0	38	154
ngk10_4	0	0	0	0	0	36	0	23	154
root	7	24	49	10866	85	70724	43169	17759	6093992
rowe	1	0	0	1	0	223	0	102	132
size	0	0	0	1	0	208	1	88	53
unix	0	0	0	1	0	68	0	44	41
xx	1	3	5	27	7	2789	60	939	58866.00

Table 6: Running times in milliseconds on the real-world graphs.

comparable results on the other criteria. It is because they create the most compact embeddings, inducing a low spread and short node movements. We can also notice on Tables 5 and 6 that *RWordle-L* can be time consuming for graphs with more than 500 nodes, which is not the case of *VPSC*. Thus, if the compactness of the embedding is one's priority, *RWordle-L* should be chosen on small graphs and *VPSC* on larger ones.

High compactness of the embeddings resulting from *RWordle-L* and *VPSC* avoids visualising the edges and the graph structures. Therefore, if one's priority is to provide an embedding highlighting the paths and the groups of nodes in the graph, the remaining options (*PFS'*, *PRISM*, *GTREE* and *Diamond*) should be favoured. Among them, *Diamond* is the slowest one with respect to the solver we used (see the introduction of this section). *Diamond* also obtains bad scores for *Node Movement minimisation* and *Edge Length preservation* on synthetic graphs (see Table 3 *nm_dm_imse* and *eb_rsdd*). *GTREE* also induces a lot of node movement, but it outperforms *Diamond* in terms of *Edge Length preservation* and running time. *PFS'* and *PRISM* obtained comparable results, outperforming *GTREE* and *Diamond* on *Node Movement minimisation*, even if *PRISM* is slightly better (see Tables 3 and 4, *nm_dm_imse*). Figures 1d and 1g illustrate their similarity while Figures 2d and 2g illustrate the ability of *PRISM* to induce less node movements on a real-world graph. *PFS'* should be favoured against *PRISM* for large graphs as its computation time is substantially lower (see Tables 5 and 6).

5 Discussion

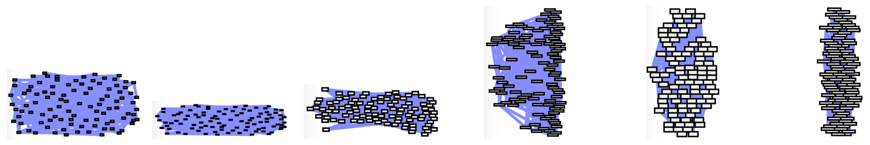
In this section, we discuss threats to validity and future research directions.

5.1 Threats to validity

Nodes aspect ratio The aspect ratio of the synthetic graphs of the above study is 2:1 whereas the aspect ratios of the real-world graphs vary with respect to the initial datasets. The fixed aspect ratio of synthetic graphs could be considered as a limitation of our study: what happens in terms of result quality among the different algorithms when the aspect ratio varies, and in particular when the width of the nodes increases to display long text labels? A clue for answering this question is available in Table 7, which shows the results for the graph of the Figure 1 with an aspect ratio of 2:1, and the same graph with an aspect ratio of 5:1. In this example, the metrics mostly highlight the same properties for both aspect ratios. The main differences appear on the global aspect ratio (*gs_bb_iar*) for *PFS*, *FTA* and *VPSC*. As illustrated by Figure 3, the drawback of spreading the embedding along one dimension is accentuated when the width of the nodes increases for *PFS* and *VPSC*. Conversely, this phenomena is attenuated for *FTA*. However, as we can see in the figure, this is due to the high movement of a bunch of nodes along the vertical axis on the right part of the embedding, which is not a sign of the output quality.

		Scaling	*PFS*	*PFS'*	*FTA*	*VPSC*	*PRISM*	*RWordle-L*	*GTREE*	*Diamond*
oo_nni	*2:1*	0.00	0.46	0.46	0.06	0.03	0.02	0.48	0.02	0.03
	5:1	0.00	0.46	0.46	0.02	0.01	0.03	0.48	0.03	0.04
sp_ch_a	*2:1*	19.52	18.51	4.96	4.28	2.53	4.63	2.04	5.97	6.80
	5:1	61.09	59.08	11.41	11.17	4.42	7.10	4.05	10.80	29.16
gs_bb_iar	*2:1*	1.06	1.87	1.23	2.48	1.97	1.29	1.03	1.19	1.11
	5:1	1.28	3.26	1.31	2.17	3.91	1.70	1.29	1.33	1.29
nm_dm_imse	*2:1*	0.00	1646.75	182.01	1173.22	210.58	196.85	464.28	302.61	553.71
	5:1	0.00	14867.75	1790.54	4549.93	298.20	630.09	2350.57	1723.94	2390.43
el_rsd	*2:1*	0.00	0.26	0.13	0.38	0.33	0.22	0.41	0.17	0.17
	5:1	0.00	0.41	0.19	0.36	0.42	0.30	0.62	0.26	0.18

Table 7: Values of the selected metrics on the graph of Figure 1 with two nodes aspect ratios: 2:1 and 5:1.



(a) *PFS* 2:1 (b) *PFS* 5:1 (c) *FTA* 2:1 (d) *FTA* 5:1 (e) *VPSC* 2:1 (f) *VPSC* 5:1

Figure 3: Overlapping-free embeddings obtained after applying *PFS*, *FTA* and *VPSC* on a graph with different nodes aspect ratios.

Number of overlaps Another factor that could limit the results is the number of overlaps of the initial embedding: are some algorithms fitted to obtain better quality measures for few or many overlaps? Table 8 shows the results obtained on the graph of Figure 1 with different sizes for the nodes, but with the same aspect ratio. The initial size, 20×10, produces 274 overlaps on the initial embedding, while the other one, 40×20, produces 1136 overlaps. Here again, the different measures mostly rank the algorithms in the same order for the two graphs. The main differences are on the aspect ratio generated by *FTA* and *VPSC*, but not by *PFS* this time. Indeed, increasing the number of overlaps accentuate the spreading along one dimension for *FTA* but not for *PFS*. *VPSC* shows the same behaviour as when we changed the aspect ratio.

5.2 Directions for future work

Further algorithms Our study focuses on algorithms explicitly designed to remove node overlaps of graph embeddings. However, a future direction could be to consider further algorithms dedicated to related problems. For instance, van Garderen *et al.* [27] propose a heuristic to remove overlaps of geo-referenced rectangles. Their objective is to minimize the displacement of the nodes while preserving the orthogonal ordering. Another example is provided by Nickel *et al.* [23] in which the authors propose a method to maintain stability among Demers time-varying cartograms, and thus provide a kind of rectangle overlap removal algorithm.

		Scaling	*PFS*	*PFS'*	*FTA*	*VPSC*	*PRISM*	*RWordle-L*	*GTREE*	*Diamond*
oo_nni	*274*	0.00	0.46	0.46	0.06	0.03	0.02	0.48	0.02	0.03
	1136	0.00	0.46	0.46	0.06	0.02	0.02	0.47	0.04	0.03
sp_ch_a	*274*	19.52	18.51	4.96	4.28	2.53	4.63	2.04	5.97	6.80
	1136	59.96	99.70	15.14	20.36	6.71	10.64	6.38	14.99	19.42
gs_bb_iar	*274*	1.06	1.87	1.23	2.48	1.97	1.29	1.03	1.19	1.11
	1136	1.13	1.72	1.17	1.95	4.65	1.33	1.25	1.02	1.10
nm_dm_imse	*274*	0.00	1646.75	182.01	1173.22	210.58	196.85	464.28	302.61	553.71
	1136	0.00	9528.27	860.05	27539.34	938.41	637.20	6547.33	1480.85	1268.12
el_rsd	*274*	0.00	0.26	0.13	0.38	0.33	0.22	0.41	0.17	0.17
	1136	0.00	0.25	0.13	0.49	0.46	0.22	0.84	0.18	0.17

Table 8: Values of the selected metrics on the graph of Figure 1 with 274 and 1136 overlaps.

Further criteria In Section 3, we described 22 metrics, classified them into 5 classes according to the properties they aim to capture and selected one of them for each class. Among the 22 metrics, 18 came from the literature on node overlap removal algorithms, the other ones were minor improvements of the previous ones based on some drawbacks highlighted in the discussions. However, we did not propose any radically different metric nor we presented how metrics coming from other applications could be adapted to meet the problem covered here. This could be an interesting future research direction but it is beyond the scope of the present work. For instance, Fadloun *et al.* [7] proposed criteria to evaluate the quality of 1D node overlap removal algorithms and it would be interesting to investigate how they could be tailored to the 2D case. Sondag *et al.* [25] proposed a refined metric to quantify the change of the relative positions of rectangles in the context of stable treemap layout algorithms. Another source of inspiration could come from geographic data visualisation. For instance, Guo and Gahegan [10] proposed several approaches to encode spatial proximity between elements and Haunert and Sering [12] quantify local distortions of road networks.

Embedding quality This study compares algorithms in regards of their ability to preserve some properties of the initial embedding. As a result, we considered only metrics that quantify how much the algorithms preserve the mental map between an initial embedding and the corresponding overlapping-free one. This approach induces a limitation that would worth future investigations beyond the scope of this paper. Indeed, there are many approaches to measure the different aspects of an embedding quality per se (see for instance [24]) and an interesting research direction would be to evaluate how much a node overlap removal algorithm degrades the quality of an initial embedding. Such a study is not trivial, as there are many layout algorithms that aim at optimizing different quality criteria, and it should be important to test embedding degradation in regards of these algorithms. For instance, consider a layout algorithm A_1 producing better results on a criterion c than another algorithm A_2, but a worse degradation of c when removing overlaps. If we want to select the best node-overlap removal algorithm to preserve c, we need to test the competitors for several initial embeddings of the same graphs resulting from different layout algorithms, and eventually select different node-overlap removal algorithms in regards of the initial layout technique employed.

6 AGORA

All the node overlap removal algorithms as well as the criteria described in the paper are available in a *JavaScript* library[12]. The implementations are those used for the experiments described in the previous section.

A Web platform, AGORA[13] (*Automatic Graph Overlap Removal Algorithms*) is also available online. The user can select one or several real-world graphs used for the experiments, or upload his/her own graphs in the GML format. The graphs must contain the nodes coordinates, x and y, and their width and height, w and h. Then he/she can select one or several node overlap removal algorithms among the nine proposed on the interface (corresponding to the ones of the experiments of the previous section). Finally, he/she can select the criteria among the 22 described above. By default, the five most relevant criteria are selected. Once these parameters have been chosen, the user can generate the overlapping-free embeddings. An embedding is provided for each graph and each algorithm, and it is shown as a thumbnail image. The user can download it as a JSON or a GML file. In this case, even if they don't appear in the thumbnail images, the node and edge properties of the original files are kept and available in the output file, just the x and y values are changed. The user can also download a thumbnail as a SVG file. Finally, a table with the values of the selected criteria for each embedding is provided.

7 Conclusion

Finding a suitable node overlap removal algorithm is difficult for a visualisation designer because even if many algorithms exist, no complete comparison based on the same criteria has been provided. In this paper we first highlighted the five main classes of existing criteria and proposed a selection of one representative criterion for each class. Using a large number of experiments carried out with synthetic and real-world graphs, we compared 9 algorithms from the state-of-the-art according to both criteria and running time. By analyzing the results, we then showed advantages, disadvantages and limitations of the algorithms, which can be very useful for the designer. Finally we proposed a *Javascript* library containing all node overlap removal algorithms and criteria as well as a Web platform, AGORA, that allows the end user to upload his/her own graphs and get the embeddings according to the selected algorithms.

[12]`https://github.com/agorajs/agorajs.github.io` (accessed: 2020-03)
[13]`https://agorajs.github.io/` (accessed: 2020-03)

References

[1] D. W. Archambault and H. C. Purchase. The "map" in the mental map: Experimental results in dynamic graph drawing. *International Journal of Human-Computer Studies*, 71(11):1044–1055, 2013. doi:10.1016/j.ijhcs.2013.08.004.

[2] A.-L. Barabaśi and R. Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, 1999. doi:10.1126/science.286.5439.509.

[3] F. Chen, L. Piccinini, P. Poncelet, and A. Sallaberry. Node overlap removal algorithms: A comparative study. In *Proceedings of the International Symposium on Graph Drawing and Network Visualization (GD)*, pages 179–192. Springer, 2019. doi:10.1007/978-3-030-35802-0_14.

[4] M. Chimani, C. Gutwenger, M. Jünger, G. W. Klau, K. Klein, and P. Mutzel. The open graph drawing framework (OGDF). In R. Tamassia, editor, *Handbook on Graph Drawing and Visualization.*, pages 543–569. Chapman and Hall/CRC, 2013.

[5] T. Dwyer, K. Marriott, and P. J. Stuckey. Fast node overlap removal. In *Proceedings of the International Symposium on Graph Drawing (GD)*, pages 153–164. Springer, 2005. doi:10.1007/11618058_15.

[6] P. Erdös and A. Rényi. On random graphs. *Publicationes Mathematicae Debrecen*, 6:290–291, 1959.

[7] S. Fadloun, P. Poncelet, J. Rabatel, M. Roche, and A. Sallaberry. Node overlap removal for 1d graph layout. In *Proceeding of the International Conference on Information Visualisation (IV)*, pages 224 – 229, 2017. doi:10.1109/iV.2017.14.

[8] E. R. Gansner and Y. Hu. Efficient, proximity-preserving node overlap removal. *Journal of Graph Algorithms and Applications*, 14(1):53–74, 2010. doi:10.7155/jgaa.00198.

[9] E. R. Gansner and S. C. North. An open graph visualization system and its applications to software engineering. *Software: practice and experience*, 30(11):1203–1233, 2000. doi:10.1002/1097-024X(200009)30:11\%3C1203::AID-SPE338\%3E3.0.CO;2-N.

[10] D. Guo and M. Gahegan. Spatial ordering and encoding for geographic data mining and visualization. *Journal of Intelligent Information Systems*, 27(3):243–266, 2006. doi:10.1007/s10844-006-9952-8.

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

[12] J. Haunert and L. Sering. Drawing road networks with focus regions. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2555–2562, 2011. `doi:10.1109/TVCG.2011.191`.

[13] K. Hayashi, M. Inoue, T. Masuzawa, and H. Fujiwara. A layout adjustment problem for disjoint rectangles preserving orthogonal order. In *Proceedings of the International Symposium on Graph Drawing (GD)*, pages 183–197. Springer, 1998. `doi:10.1007/3-540-37623-2\_14`.

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

[15] X. Huang and W. Lai. Force-transfer: a new approach to removing overlapping nodes in graph layout. In *Proceedings of the Australasian Computer Science Conference (ACSC)*, pages 349–358, 2003.

[16] X. Huang, W. Lai, A. Sajeev, and J. Gao. A new algorithm for removing node overlapping in graph visualization. *Information Sciences*, 177(14):2821 – 2844, 2007. `doi:10.1016/j.ins.2007.02.016`.

[17] W. Li, P. Eades, and N. Nikolov. Using spring algorithms to remove node overlapping. In *Proceedings of the Asia-Pacific Symposium on Information Visualisation (APVis'05)*, pages 131–140, 2005.

[18] K. A. Lyons, H. Meijer, and D. Rappaport. Algorithms for cluster busting in anchored graph drawing. *Journal of Graph Algorithms and Applications*, 2(1):1–24, 1998. `doi:10.7155/jgaa.00004`.

[19] K. Marriott, P. Stuckey, V. Tam, and W. He. Removing node overlapping in graph layout using constrained optimization. *Constraints*, 8(2):143–171, 2003. `doi:10.1023/A:1022371615202`.

[20] W. Meulemans. Efficient optimal overlap removal: Algorithms and experiments. *Computer Graphics Forum*, 38(3):713–723, 2019. `doi:10.1111/cgf.13722`.

[21] K. Misue, P. Eades, W. Lai, and K. Sugiyama. Layout adjustment and the mental map. *Journal of Visual Languages & Computing*, 6(2):183–210, 1995. `doi:10.1006/jvlc.1995.1010`.

[22] L. Nachmanson, A. Nocaj, S. Bereg, L. Zhang, and A. Holroyd. Node overlap removal by growing a tree. In *Proceedings of the International Symposium on Graph Drawing and Network Visualization (GD)*, pages 33–43. Springer, 2016. `doi:10.1007/978-3-319-50106-2\_3`.

[23] S. Nickel, M. Sondag, W. Meulemans, M. Chimani, S. G. Kobourov, J. Peltonen, and M. Nöllenburg. Computing stable demers cartograms. In *Proceedings of the International Symposium on Graph Drawing and Network Visualization (GD)*, pages 46–60. Springer, 2019. `doi:10.1007/978-3-030-35802-0\_4`.

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

[25] M. Sondag, B. Speckmann, and K. Verbeek. Stable treemaps via local moves. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):729–738, 2018. doi:10.1109/TVCG.2017.2745140.

[26] H. Strobelt, M. Spicker, A. Stoffel, D. A. Keim, and O. Deussen. Rolled-out wordles: A heuristic method for overlap removal of 2D data representatives. *Computer Graphics Forum*, 31(3):1135–1144, 2012. doi:10.1111/j.1467-8659.2012.03106.x.

[27] M. van Garderen, B. Pampel, A. Nocaj, and U. Brandes. Minimum-displacement overlap removal for geo-referenced data visualization. *Computer Graphics Forum*, 36(3):423–433, 2017. doi:10.1111/cgf.13199.

[28] D. J. Watts and S. H. Strogatz. Collective dynamics of ‘small-world’ networks. *Nature*, 393:440–442, 1998. doi:10.1038/30918.