Node Overlap Removal Algorithms: A Comparative Study

Fati Chen[1,2(✉)], Laurent Piccinini[2], Pascal Poncelet[1], and Arnaud Sallaberry[1,2]

[1] LIRMM - CNRS - Université de Montpellier, Montpellier, France
{fati.chen,pascal.poncelet,arnaud.sallaberry}@lirmm.fr
[2] Université Paul-Valéry Montpellier 3, Montpellier, France
{fati.chen,laurent.piccinini,arnaud.sallaberry}@univ-montp3.fr

Abstract. 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 the quality has never been provided and it is thus difficult for a visualization designer to select the algorithm that best suits his needs. In this paper, we review 21 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 8 node overlap removal algorithms. Our experiment involves 854 synthetic and real-world graphs.

Keywords: Graph drawing · Layout adjustment · Node overlap removal

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 graphs, it causes node overlap which hides information. Post-process algorithms, named *layout adjustment* [15], 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 of the original embedding. Thus, it is also important to minimise the change on the layout.

Since a preliminary work in 1995 [15], many algorithms have been designed to reach these purposes, and many quality criteria have been proposed to evaluate

D. Archambault and C. D. Tóth (Eds.): GD 2019, LNCS 11904, pp. 179–192, 2019.
https://doi.org/10.1007/978-3-030-35802-0_14

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 two forms: (1) We propose a classification of 21 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 overlapping approaches in regards to the previously selected metrics. This experiment involves 854 graphs, including synthetic ones (random, tree, scale-free, small-world) and real world ones.

The paper is organised as follows: after a brief reminder in Sect. 2 of the definitions and the notations used in this paper, we present and discuss the quality criteria and the metrics in Sect. 3. Then we compare the algorithms in Sect. 4. Finally we conclude in Sect. 5.

2 Preliminaries

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

$G = (V, E)$ denotes a graph where V is the set of nodes and E the 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 impacted by the layout adjustment.

The initial embedding is defined as an injection $\mathcal{E}_G : V \rightarrow \mathbb{R}^2$ 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. With a view 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*, *Spread minimisation*, *Global Shape preservation*, *Node Movement minimisation* and *Edge Length preservation*), each of them depicting a quality criterion. Table 1 shows the metrics assigned into the classes. The formulas are given in the discussion below.

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, Sect. 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. We recall that the orthogonal ordering is respected when all nodes satisfy 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, *oo_o* [15], 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.* [11] proposed a metric based on the *Kendall's Tau distance*. For each couple of nodes, they first compute an inversion number $inv(u, v)$ corresponding to 0 if the orthogonal ordering is preserved

Table 1. List of metrics classified by the quality criteria they try to capture: metrics selected for the comparison appear in bold italics. The *Abbreviations* are based on some initials of the names. For example, *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 *Target* column refers to the target value to meet the corresponding criterion.

Abbreviation	Name	Range	Target
	Orthogonal Ordering preservation		
oo_o	Original [15]	$\{0, 1\}$	1
oo_kt	Kendall's Tau Distance [11]	$[0, 1]$	0
oo_ni	Number of Inversions [17]	$[0, n(n-1)]$	0
oo_nni	***Normalised Number of Inversions***	$[0, 1]$	0
	Spread minimisation		
sp_bb_l1ml	Bounding Box L1 Metric Length [12]	$[1, +\infty[$	1
sp_bb_a	Bounding Box Area [15]	$[1, +\infty[$	1
sp_bb_na	Bounding Box Normalised Area [11]	$[0, 1[$	0
sp_ch_a	***Convex Hull Area*** [17]	$[1, +\infty[$	1
	Global Shape preservation		
gs_bb_ar	Bounding Box Aspect Ratio [12]	$]0, +\infty[$	1
gs_bb_iar	***Bounding Box Improved Aspect Ratio***	$[1, +\infty[$	1
gs_ch_sd	Convex Hull Standard Deviation [17]	$[0, +\infty[$	0
	Node Movement minimization		
nm_mn	Moved Nodes [11]	$[0, 1]$	0
nm_dm_me	Distance Moved Mean Euclidean [17]	$[0, +\infty[$	0
nm_dm_ne	Distance Moved Normalized Euclidean [13]	$[0, 1]$	0
nm_dm_h	Distance Moved Hamiltonian [10,11]	$[0, +\infty[$	0
nm_dm_se	Distance Moved Squared Euclidean [14]	$[0, +\infty[$	0
nm_dm_imse	***Distance Moved Improved Mean Squared Euclidean***	$[0, +\infty]$	0
nm_d	Displacement [5]	$]0, +\infty[$	0
nm_knn	K-Nearest Neighbours [16]	$[0, +\infty[$	0
	Edge Length preservation		
el_r	Ratio [12]	$[1, +\infty[$	1
el_rsdd	***Relative Standard Deviation Delaunay*** [5]	$[0, +\infty]$	0

between them, 1 otherwise. The metric is then defined as the normalised sum of the inversion numbers:

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

Strobelt *et al.* [17] introduced the number of inversions:

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

This metric has the drawback of providing non-normalized 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 [12] 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 [15] 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.* [11] 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 poorly intuitive 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 [17], 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.* [12]:

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

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 $w_{bb} = 3$, $h_{bb} = 2$, $w'_{bb} = 6$, $h'_{bb} = 4$. 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 $w_{bb} = 3$, $h_{bb} = 2$, $w'_{bb} = 4$, $h'_{bb} = 6$. 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. The main drawback of this metric is that it can reach values in the interval $]0, +\infty[$ while the target value is 1. Thus, it is hard to decide, for instance, which algorithm is the best between two of them if the first one obtains a score of 0.67 and the second one a score of 4.56. 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. 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.* [17]. 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 Sect. 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.* [11]:

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

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.

Table 2. Functions used to tune the distances moved metric.

$\text{dist}(v, v') \quad \backslash \quad f(n)$	1	$1/n$	$\frac{1}{k\sqrt{2}\times n}$
$\|v' - v\|$		*nm_dm_me* [17]	*nm_dm_ne* [13]
$\|v' - v\|^2$	*nm_dm_se* [14]	*nm_dm_imse*	
$\|x'_v - x_v\| + \|y'_v - y_v\|$	*nm_dm_h* [10]		

The function f comes in three different forms. Marriott *et al.* [14] and Huang *et al.* [10] 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.* [17] proposed to use the mean of the

distances, which corresponds to $f(n) = 1/n$. Finally, Lyons *et al.* [13] 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. Thus, this f function normalises the values of the metric. Unfortunately, this normalisation induces very small values that are hard to interpret. That is why we preferred using $f(n) = 1/n$ for our study.

Three *dist* functions have been proposed in the literature. The most intuitive one is the Euclidean distance $\|v' - v\|$ [13,17]. The squared Euclidean distance $\|v' - v\|^2$ [14] avoids the square root computation and discriminates high changes better. It is the one we selected for our study. The Manhattan distance $|x'_v - x_v| + |y'_v - y_v|$ has also been used [10], 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:

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

Considering this, we selected the following node movement metric:

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

nm_d [5] (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 minimizes 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 inside 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 KNN approach, Nachmanson *et al.* [16] proposed the following metric:

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

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 [12]:

$$el\_r = \frac{\max_{(u,v)\in E^2} \|u' - v'\|}{\min_{(u,v)\in E^2} \|u' - v'\|}$$

While many layout algorithms are not designed to produce uniform edge lengths, we did not select this approach, which is not related to the mental model preservation for these algorithms. We preferred the next one, based on 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 [5]:

$$r_{uv} = \frac{||u' - v'||}{||u - v||}, \quad (u, v) \in E_{dt}^2$$

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

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

4 Algorithms Comparison

In this section, we compare 8 algorithms of the literature in terms of quality and running time: uniform *Scaling*, *PFS* [15], *PFS'* [8], *FTA* [11], *VPSC* [3], *PRISM* [5], *RWordle-L* [17], and *GTREE* [16]. The quality of an overlapping-free embedding is evaluated with the metrics identified in the last section, by following a 3 steps procedure: **Step 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 [2]: random graphs [4], random trees, small world graphs [18], and scale-free graphs [1]. We also use 14 real-world graphs selected

from the Graphviz test suite[1] [6], previously used by the authors of *PRISM* [5] and *GTREE* [16]. **Step 2: Overlapping-free embedding computation** Synthetic graphs resulting from the first step are initially positioned by the FM^3 layout algorithm [7]. Then, we apply the 8 node overlap removal algorithms, thus providing a set of 6,720 overlapping-free graph embeddings. `Graphviz` test suite graphs are initially positioned by the *SFDP* layout algorithm [9] to follow the same baseline embedding as Gansner *et al.* [5]. We then apply the 8 node overlap removal algorithms thus providing 112 overlapping-free graph embeddings. **Step 3: Metrics computation** We finally compute the values of the 5 selected metrics on the 6.832 overlapping-free synthetic and real-world graph embeddings. We also measure the computation time of the algorithms.

4.1 Quality

Figure 1 and 2 show the aggregated metrics values on the synthetic and real-world datasets. 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.

		Scaling	*PFS*	*PFS'*	*FTA*	*VPSC*	*PRISM*	*RWordle-L*	*GTREE*
oo_nni	*Q1*	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	Median	0.00	0.00	0.00	0.01	0.00	0.01	0.01	0.02
	Q3	0.00	0.00	0.00	0.04	0.03	0.02	0.10	0.03
sp_ch_a	*Q1*	1.96	1.04	1.00	1.00	1.00	1.01	1.00	1.12
	Median	8.70	1.69	1.22	1.02	1.00	1.12	1.01	1.49
	Q3	34.29	17.63	5.04	4.21	2.30	4.22	1.99	5.94
gs_bb_iar	*Q1*	1.00	1.01	1.00	1.00	1.00	1.00	1.00	1.01
	Median	1.01	1.19	1.04	1.01	1.00	1.04	1.00	1.04
	Q3	1.06	1.65	1.14	1.66	1.94	1.23	1.07	1.08
nm_dm_imse	*Q1*	0.00	6.50	1.65	0.94	0.42	9.82	2.18	28.22
	Median	0.00	694.83	116.06	37.87	9.71	131.57	27.60	292.56
	Q3	0.00	7899.2	1782.8	2966.0	283.6	688.1	597.7	2221.7
eb_rsdd	*Q1*	0.00	0.05	0.05	0.04	0.03	0.13	0.07	0.13
	Median	0.00	0.25	0.21	0.22	0.17	0.31	0.25	0.28
	Q3	0.00	0.33	0.26	0.60	0.47	0.38	0.68	0.36

Fig. 1. Aggregated values of the selected metrics among the synthetic graphs: first quartile, median and third quartile.

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.

[1] https://gitlab.com/graphviz/graphviz/blob/master/rtest/graphs/ (accessed: 2019-07).

	Scaling	*PFS*	*PFS'*	*FTA*	*VPSC*	*PRISM*	*RWordle-L*	*GTREE*
oo_nni	0.00	0.00	0.00	0.07	0.01	0.02	0.04	0.02
sp_ch_a	217.26	210.28	6.92	6.03	2.36	2.18	1.53	4.02
gs_bb_iar	1.04	1.97	1.22	1.95	2.20	1.33	1.04	1.17
nm_dm_imse	0.00	9768157.94	63594.74	4297355.84	34611.40	42919.66	35928.91	37331.83
eb_rsdd	0.00	0.39	0.28	0.77	0.34	0.37	0.50	0.35

Fig. 2. Mean values of the selected metrics among the real-world graphs.

Scaling preserves the initial global shape[2] (*gs_bb_iar* score). *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.

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. Finally, *GTREE* had bad results on the synthetic graphs, while it obtained pretty good ones on the real-world graphs.

Scaling preserves relative edge lengths. All the other criteria obtained comparable median score between 0.17 and 0.31. However, the third quartile on the synthetic graphs shows that *FTA*, *VPSC* and *RWordle-L* generate a certain amount of embeddings with high variations. This observation is confirmed by the results on the real-world graphs for *FTA* and *RWordle-L*.

4.2 Computation Time

Figure 3 and 4 show the aggregated running time values on the synthetic and real-world datasets. *Scaling*, *PFS*, *PFS'* and *VPSC* require lower running time. *FTA* and *GTREE* induce intermediate running time, but the third quartile shows that *FTA* can induce a certain amount of time consuming embedding computations. Finally, *PRISM* is time consuming for small graphs, but have intermediate results for larger graphs, while *RWordle-L* has good results for small graphs but is very time-consuming for larger ones.

[2] The global shape preservation score for *Scaling* is not 1 because of the size of the nodes that remains the same between the initial and the overlapping-free embeddings.

		Scaling	PFS	PFS'	FTA	VPSC	PRISM	RWordle-L	GTREE
10	Q1	0.00	0.00	0.00	0.00	0.00	3.00	0.00	0.00
	Median	0.00	0.00	0.00	0.00	0.00	4.00	0.00	1.00
	Q3	0.00	0.00	0.00	0.00	0.00	12.00	0.00	3.00
20	Q1	0.00	0.00	0.00	0.00	0.00	4.00	0.00	1.00
	Median	0.00	0.00	0.00	0.00	0.00	6.00	0.00	1.00
	Q3	0.00	0.00	0.00	0.00	1.00	9.00	1.00	4.00
50	Q1	0.00	0.00	0.00	0.00	1.00	7.00	0.00	2.00
	Median	0.00	0.00	0.00	1.00	1.00	16.00	3.00	5.00
	Q3	0.00	0.00	1.00	2.00	2.00	41.25	9.25	8.25
100	Q1	0.00	0.00	1.00	1.00	2.00	25.75	0.00	8.00
	Median	0.00	1.00	1.00	4.00	3.00	44.00	36.50	13.00
	Q3	1.00	1.00	1.00	12.00	4.00	69.00	84.25	26.25
200	Q1	1.00	2.00	4.00	4.00	4.00	40.75	2.00	19.00
	Median	1.00	2.00	4.00	25.50	10.50	81.50	274.00	30.00
	Q3	2.00	3.00	5.00	89.00	15.00	134.25	589.50	47.00
500	Q1	2.00	15.00	25.00	26.00	25.00	196.75	21.00	59.50
	Median	10.00	17.00	29.50	207.50	93.00	308.00	3410.00	109.00
	Q3	13.00	20.00	35.00	1392.00	140.75	487.25	7246.25	174.75
1000	Q1	4.75	57.00	113.50	87.75	125.50	623.75	112.00	165.00
	Median	34.50	67.50	133.00	1022.00	496.00	1053.00	26877.50	297.00
	Q3	58.25	77.00	176.25	8694.00	1065.75	1440.00	55173.25	418.75

Fig. 3. Aggregated running times among 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
b100	41	82	126	19823	321	3750	191109	285
b102	5	16	24	82	15	229	96	33
b124	0	0	0	3	1	28	1	6
b143	0	0	1	9	2	42	6	11
badvoro	20	43	63	32553	383	1512	56155	213
dpd	0	0	0	1	1	1	0	1
mode	0	1	2	93	4	172	281	34
NaN	0	1	1	2	0	17	0	7
ngk10_4	0	0	0	0	0	9	0	2
root	7	24	49	10866	85	2301	43169	192
rowe	1	0	0	1	0	3	0	1
size	0	0	0	1	0	18	1	2
unix	0	0	0	1	0	5	0	1
xx	1	3	5	27	7	158	60	35

Fig. 4. Mean values of running times among the real-world graphs.

5 Conclusion

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. *FTA* obtained intermediate results over all the criteria, which is less good than all its remaining competitors. *PFS'* and *PRISM* obtained comparable results but the latter is more time-consuming. Both

have intermediate results for shape preservation and node movement minimisation, which might be considered as two essential criteria. *GTREE* suffers from inducing high node movements on our datasets. Overall, *VPSC* and *RWordle-L* obtained the best quality results. While *RWordle-L* outperforms *VPSC* on global shape preservation and is comparable on the other criteria, *VPSC* outperforms *RWordle-L* in terms of running time. Finally, considering the different types of graphs (random graphs, random trees, small world graphs, and scale-free graphs), we did not observe any significant differences in terms of results.

Acknowledgement. This research has been partly funded by a national French grant (ANR Daphne 17-CE28-0013-01).

References

1. Barabaśi, A.L., Albert, R.: Emergence of scaling in random networks. Science **286**(5439), 509–512 (1999)
2. Chimani, M., Gutwenger, C., Jünger, M., Klau, G.W., Klein, K., Mutzel, P.: The open graph drawing framework (OGDF). In: Tamassia, R. (ed.) Handbook on Graph Drawing and Visualization, pp. 543–569. Chapman and Hall/CRC, London (2013)
3. Dwyer, T., Marriott, K., Stuckey, P.J.: Fast node overlap removal. In: Healy, P., Nikolov, N.S. (eds.) GD 2005. LNCS, vol. 3843, pp. 153–164. Springer, Heidelberg (2006). https://doi.org/10.1007/11618058_15
4. Erdös, P., Rényi, A.: On random graphs. Publicationes Mathematicae Debrecen **6**, 290–291 (1959)
5. Gansner, E., Hu, Y.: Efficient, proximity-preserving node overlap removal. J. Graph Algorithms Appl. **14**(1), 53–74 (2010)
6. Gansner, E.R., North, S.C.: An open graph visualization system and its applications to software engineering. Softw. Pract. Exp. **30**(11), 1203–1233 (2000)
7. Hachul, S., Jünger, M.: Drawing large graphs with a potential-field-based multilevel algorithm. In: Pach, J. (ed.) GD 2004. LNCS, vol. 3383, pp. 285–295. Springer, Heidelberg (2005). https://doi.org/10.1007/978-3-540-31843-9_29
8. Hayashi, K., Inoue, M., Masuzawa, T., Fujiwara, H.: A layout adjustment problem for disjoint rectangles preserving orthogonal order. In: Whitesides, S.H. (ed.) GD 1998. LNCS, vol. 1547, pp. 183–197. Springer, Heidelberg (1998). https://doi.org/10.1007/3-540-37623-2_14
9. Hu, Y.: Efficient, high-quality force-directed graph drawing. Math. J. **10**(1), 37–71 (2005)
10. Huang, X., Lai, W.: Force-transfer: a new approach to removing overlapping nodes in graph layout. In: Proceedings of the 26th Australasian computer science conference, vol. 16, pp. 349–358. Australian Computer Society, Inc. (2003)
11. Huang, X., Lai, W., Sajeev, A., Gao, J.: A new algorithm for removing node overlapping in graph visualization. Inf. Sci. **177**(14), 2821–2844 (2007)
12. Li, W., Eades, P., Nikolov, N.: Using spring algorithms to remove node overlapping. In: Proceedings of the 2005 Asia-Pacific Symposium on Information Visualisation, vol. 45, pp. 131–140. APVis 2005. Australian Computer Society Inc, Darlinghurst (2005)
13. Lyons, K.A., Meijer, H., Rappaport, D.: Algorithms for cluster busting in anchored graph drawing. J. Graph Algorithms Appl. **2**(1), 1–24 (1998)

14. Marriott, K., Stuckey, P., Tam, V., He, W.: Removing node overlapping in graph layout using constrained optimization. Constraints **8**(2), 143–171 (2003)
15. Misue, K., Eades, P., Lai, W., Sugiyama, K.: Layout adjustment and the mental map. J. Vis. Lang. Comput. **6**(2), 183–210 (1995)
16. Nachmanson, L., Nocaj, A., Bereg, S., Zhang, L., Holroyd, A.: Node overlap removal by growing a tree. In: Hu, Y., Nöllenburg, M. (eds.) GD 2016. LNCS, vol. 9801, pp. 33–43. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-50106-2_3
17. Strobelt, H., Spicker, M., Stoffel, A., Keim, D., Deussen, O.: Rolled-out wordles: a heuristic method for overlap removal of 2D data representatives. Comput. Graph. Forum **31**(3), 1135–1144 (2012)
18. Watts, D.J., Strogatz, S.H.: Collective dynamics of 'small-world' networks. Nature **393**, 440–442 (1998)