Treecapitator

A new power law graph 3D layout algorithm

NexusPrime

January 29, 2026

Abstract

Rendering real networks as well as graphs that exhibit small world characteristics (primarly a power law degree sequence) can be quite challenging. Current state-of-the art layout algorithms fail at placing the nodes in a meaningful readable way. Typical strategies create a big agglomeration of nodes and edges in the center that makes the understanding of the visualization very difficult. In this work I explore an alternative method for placing the nodes based on their node degree. This method is not only composed of the numerical placement of the nodes in the network but also of a rendering algorithm that is capable of providing useful coloring and alpha blending to the result, making it easy to infer network structure and connection density even in the most cluttered networks.

Contents

1 Introduction **2**
- 1.1 Research Questions . . . 2
- 1.2 Experimentation . . . 2

2 A new power law model **3**
- 2.1 Randomization . . . 3
- 2.2 Statistical analysis of node degree . . . 3

3 A new 3D layout method **4**
- 3.1 Method definition . . . 4
  - 3.1.1 Preliminaries . . . 4
  - 3.1.2 Computation of A(u) . . . 4
  - 3.1.3 Computation of B(u) . . . 4
  - 3.1.4 Computation of C(u) . . . 4
  - 3.1.5 Position blending . . . 4
  - 3.1.6 Optimization . . . 5
- 3.2 Node colouring . . . 5
- 3.3 Heuristic Rationale . . . 5
  - 3.3.1 A(u) . . . 5
  - 3.3.2 B(u) . . . 5
  - 3.3.3 C(u) . . . 5
  - 3.3.4 Blending . . . 6

4 Results **6**
- 4.1 Synthetic model graph comparison . . . 6
  - 4.1.1 Effects on prefferential attachment power-law exponent . . . 6
  - 4.1.2 Effects on Watts–Strogatz rewiring probability . . . 6
  - 4.1.3 Erdős–Rényi variable connectivity probability . . . 6
- 4.2 Natural networks comparison . . . 6
- 4.3 Layout parameter effects comparison . . . 6
  - 4.3.1 Parameter b effect . . . 6
  - 4.3.2 Parameter c effect . . . 7
  - 4.3.3 Parameter τ effect . . . 7
- 4.4 Power-Law new model statistical measures 7
- 4.5 Rendering parameters effect comparison . . 7

5 Discussion **7**
- 5.1 Validity of the new power-law model . . . 7
- 5.2 Comparison of the artifical models . . . 7
  - 5.2.1 Variation of γ exponent in Prefferential Attachment . . . 8
  - 5.2.2 Variation in rewiring of Watts–Strogatz model . . . 8
  - 5.2.3 Variation of probability in Erdós–Rényi model . . . 8
- 5.3 Comparison of the natural models . . . 9
  - 5.3.1 Email network . . . 9
  - 5.3.2 Road network . . . 9
  - 5.3.3 Google-WebGraph . . . 9
  - 5.3.4 Linguistic Graph . . . 9
  - 5.3.5 Summary . . . 9
- 5.4 Effects of the layout parameters . . . 10
  - 5.4.1 Effect of the parameter b . . . 10
  - 5.4.2 Effect of the parameter c . . . 10
  - 5.4.3 Effect of the parameter τ . . . 10
- 5.5 Effects of the rendering parameters . . . 11
  - 5.5.1 Effect of the colouring metric . . . 11
  - 5.5.2 Effect of alpha blending and line thickness . . . 12

6 Conclusions **12**
- 6.1 RQ1 . . . 12
- 6.2 RQ2 . . . 12

6.3 RQ3 12
6.4 RQ4 13
6.5 RQ5 13
6.6 Final conclusion 13

7 Methods **13**
7.1 Choice of parameters and experiments . . . 13
7.2 Synthetic networks 14
7.3 Data Validation 14
7.4 Power-law degree sequence plot 14
7.5 Implementation Note 15

1. Introduction

Drawing power law degree sequence graph is a challenging task for common algorithms, typical layouts agglomerate most nodes in the middle, fot that matter I have developed an alternative method keeping in mind to rapidly move away the higher degree holders of the graph and placing each node of the network in reverse degree order.

For the development of the project a new power-law model has been created with the aim of providing a fast source of synthetic data to test the layout algorithm against. This algorithm as opposed to the traditional power-law models based on network growth and prefferential attachment works only with an exact function evaluation to test edge connectivity between 2 nodes in O(1).

Additionaly the rendering algorithm will take the 3D layout generated and position it in a 3D render scene with OpenGL, this gives more advantages both in the degree of rendering freedom and performance allowing to render very large networks (the order of millions of nodes) easily.

To enhance the visualization a new method of for coloring both nodes and edges have been proposed.

1.1. Research Questions

These are the main research questions for testing my proposed algorithm:

- **RQ1** Does the presented layout offer a visually clear representation for the power-law degree graphs?
- **RQ2** Is the representation useful?
- **RQ3** What is the behaviour with other types of graphs that do not follow a power-law degree sequence?
- **RQ4** Does the layout offer a representation improvement over natural graphs?
- **RQ5** How do the parameters change the behaviour? Are they equally relevant or necessary? Are they graph dependent? Can they be fine tuned?

1.2. Experimentation

In this section I will describe the experiments that I will perform to give answer to the *Research Questions*.

These are the sources of the network models the algorithm will be tested against, coming from Stanford Network Analysis Project (2016) and according to them:

- *email-EuAll*: The network was generated using email data from a large European research institution. For a period from October 2003 to May 2005 (18 months), given a set of email messages, each node corresponds to an email address. We create a directed edge between nodes i and j, if i sent at least one message to j.
- *Google-WebGraph*: Nodes represent web pages and directed edges represent hyperlinks between them.

- *roadNet-CA(California road network)*: A road network of California. Intersections and endpoints are represented by nodes and the roads connecting these intersections or road endpoints are represented by undirected edges.
- *Linguistic catalan graph*: This graph is obtained by first taking the syntactic dependency tree of Universal Dependencies Nivre and et al. (2016) tree bank of Catatalan in the AnCora corpus(Taulé et al. (2008)). To obtain the augmented graph I have mapped every typed found in the respective conllu file with a node, each pair of nodes is connected if there is any syntactical connection between 2 of their tokens.

Net.	N	E	$\langle k \rangle$
Email-EU (dir.)	265k	420k	1.58
Google-Web (dir.)	876k	5.1M	5.83
RoadNet-CA (undir.)	2.0M	2.8M	2.82
Catalan Ling. (undir.)	7.2k	57.7k	15.91

Table 1: Basic statistics of analyzed networks.

In parallel to testing with this networks, I will test the default layout algoirithm in porgram Graphia Graphia Developers (2024) which is highly efficient graph rendering utility capable of visualizing large networks. The rationale of this choice is:

- It renders the nodes directly and edges directly as opposed to other mainstream algoirithms that simplify large power-law degree based graphs.
- It generates a 3D layout which is more suitable of comparison with my layout algorithm which is also 3D.
- Applies a modern set of heuristics and Force Directed optmization making it suitable for a variety of large graphs with different structures.

Additionally the algorithm will be tested against (Erdós–Rényi, Barabási–Albert, Watts–Strogatz, Factor Graph(the new Power-Law model)). The algoirithm will be tested against:

- Varying degree of p probability in Erdós–Rényi model.
- Varying degree of rewiring probability β in Watts–Strogatz.
- Varying degree of exponent γ in Barabási–Albert.

2. A new power law model

In this section I will provide the definition of the new power law model (I decided to call Factor Graph) developed, starting from the traditional definition of graph:

$$G = (V, E).$$

We first need to establish a labelling function that maps each vertex to a natural number from a continous range over $[2, |V|]$.

$$\ell : V \xrightarrow{\sim} \{2, 3, 4, \ldots, |V|\}.$$

- We associate to each node of the graph a natural number such that each node is labelled consecutively and starting by 2
- For the edge collection we follow the nomeclature that an edge of the graph is (u,v) such that the associated numbers to the nodes $l(u) > l(v)$. This will be important for later on.

From this point on we will work with the labelling of the node instead of its typical index.

Then we define the connectivity rule for each pair of vertices:

$$E \subset \left\{ (u, v) \;\middle|\; \exists\, k \in \mathbb{N} \text{ such that } u = k\, v^{\gamma} \right\}.$$

Where gamma is a constant specified by us, this rule essentially is telling is that every pair of nodes (u, v) is connected if either u divides v or v divides u.

2.1. Randomization

We can induce randomization to the generation of the network by adding a hashing rule (a random bijective mapping) to the generation of the graph such:

$$E \subset \left\{ (u, v) \;\middle|\; \exists\, k \in \mathbb{N} \text{ such that } hash(u) = k\, v^{\gamma} \right\}.$$

This will randomize the endpoints of one side of the connection, it keeps the original small world structure with different labeling.

2.2. Statistical analysis of node degree

We can analyse what the average degree of given number is for a large enough value of $|V|$. For every vertex u, we can divide the set of connections of a given node u in 2 sets:

$$L(u) = \left\{ v \;\middle|\; \exists\, k \text{ such that } u = k\, v^{\gamma} \right\}.$$

$$R(u) = \left\{ v \;\middle|\; \exists\, k \in \mathbb{N} \text{ such that } v = k\, u^{\gamma} \right\}.$$

For computing the size of $R(u)$ we have to look for the number of values that have u^{γ} as divider not exceeding $|V|$, that is on average:

$$|R(u)| = \left\lfloor |V| \cdot u^{-\gamma} \right\rfloor$$

For computing the expected size of $L(u)$, we consider the average over all vertices. Let U be a uniformly random vertex label in $\{1, \ldots, |V|\}$. We define the indicator random variables

$$X_i = \mathbf{1}_{\{\, \exists\, k \text{ such that } U = k\, i^{\gamma} \,\}}.$$

Then the probability that i^{γ} divides U is given by

$$\mathbb{P}(X_i = 1) = \frac{\lfloor |V| / i^{\gamma} \rfloor}{|V|} = \frac{1}{i^{\gamma}} + o(1).$$

Therefore, the expected size of $L(U)$ is

$$\mathbb{E}[|L(u)|] = \sum_{i \le |V|^{1/\gamma}} \mathbb{E}[X_i] \sim \sum_{i=1}^{\infty} \frac{1}{i^{\gamma}}.$$

In particular, for $\gamma > 1$ this sum converges and we obtain

$$\mathbb{E}[|L(u)|] = O(\zeta(\gamma)).$$

On average, the degree of a vertex satisfies

$$\mathbb{E}[\deg(u)] = \mathbb{E}[|L(u)|] + \mathbb{E}[|R(u)|] = O(\zeta(\gamma)) + |V| \cdot U^{-\gamma}.$$

3. A new 3D layout method

In this section I will explain the new method being proposed, this method takes the graph and generates both the positioning and the colours for the nodes of the network, additionally the method is enhanced with a force directed based method optimizer that helps distributing the higher degree holders to avoid undesired overlapped caused by randomization artifacts.

Additionally during this process the colour of the node is computed taking a similar rule to the position one.

3.1. Method definition

The method operates heuristically looking only at each node degree and immediate connections one at a time. We first sort all nodes in the network in descendent node degree order, for each node we compute positions $A(u), B(u), C(u)$, then the method performs a blending of the 3 and gives the final position of the node.

3.1.1 Preliminaries

Vector parameters

We define some basic variables of a node and vectors of sets of nodes:

- Let d_i be the degree of node i.
- Let p_i be the position of node i.
- For a node set $S = \{i_1, i_2, \ldots, i_{|S|}\}$ with a fixed ordering, define

  $$d(S) = \big(d_{i_1}, d_{i_2}, \ldots, d_{i_{|S|}}\big)$$

  as the vector of degrees of the nodes in S, ordered according to the chosen indexing of S.
- Similarly, define

  $$p(S) = \big(p_{i_1}, p_{i_2}, \ldots, p_{i_{|S|}}\big)$$

  as the vector of positions of the nodes in S, ordered consistently with $d(S)$.

Softmin

$$\tilde{w}_i = \exp\left(-\frac{x_i - \min_j x_j}{\tau}\right), \qquad i = 1, \ldots, n$$

$$\mathrm{softmin}_{\tau}(\mathbf{x})_i = \frac{\tilde{w}_i}{\sum_{k=1}^{n} \tilde{w}_k}$$

For a given value of τ

Randdir

We define function $randdir()$ which returns a random unit 3D vector.

3.1.2 Computation of A(u)

We first need to compute set $R(u)$ from the node adjacency list $N(u)$ such that we are only interested in the nodes with higher degree than u.

$$R(u) = \big\{\, v \,\big|\, v \in N(u) \wedge d_v > d_u \big\}.$$

Then from this set we compute:

$$A(u) = \mathrm{softmin}\big(D(R(u))\big)^{\top} P(R(u))$$

Notice that all positions of set $R(u)$ have already been computed by previous iterations of the method.

3.1.3 Computation of B(u)

Computing $B(u)$ is basically computing a random direction vector and then multiplying by a parameter of the method, b.

$$B(u) = randdir() \cdot b$$

3.1.4 Computation of C(u)

Computing $C(u)$ is basically computing a random direction vector and then multiplying by a parameter of the method, c.

$$C(u) = randdir() \cdot c$$

3.1.5 Position blending

Finally the final position is computed by:

$$P(u) = B(u) \cdot l(u) + A(u) \cdot (1 - l(u)) + C(u)$$

Where $l(u)$ is:

$$l(u) = \frac{d(u)}{\max(D(V))}$$

The final result is a linear interpolation over parameter $l(u)$ of $A(u)$ and $B(u)$ with a random jitter $C(u)$ that can be adjusted depending on the graph.

From this algorithm we have the following tunnable parameters:

- τ: This parameter controls how greedy the selection intent of the positioning of $A(u)$ is.
- b: This parameter controls the influence of $B(u)$
- c: This parameter controls the influence of $C(u)$

From the definition $b >> c$.

3.1.6 Optimization

This is an optional step that is given to correct the positions of the higher degree holders (in their case $B(u)$ dominates the position) which can make some hubs appear really close although they have no connections or are unrelated to. This is primarly an issue in networks that are highly structured.

To solve this I use the following method which is an adaptation of a Force-Directed based algorithm using Euler iterative method.

Node selection

We first select the nodes that we are going to optimize simply by:

$$H = \left\{ u \,\middle|\, d_u > max(D(V)) \cdot h \right\}$$

being h a value between 0 and 1 that is to select the range of nodes to be chosen.

Method iteration

Let:

- $p_i(u)$: Position of of node u at iteration i.
- $v_i(u)$: Velocity of of node u at iteration i.
- $p_0(u)$: Initial position of node u (being $B(u)$).
- $v_0(u)$: Initial Velocity of node u (being $(0, 0, 0)$).
- $f_i(u)$: Force of of node u at iteration i.
- dt : Time between itertations.

We first compute the force for each node:

$$f_i(u) = \sum_{v \in H - \{u\}} k \cdot \frac{d_u \cdot d_v}{max(D(V))^2 \cdot |p_{i-1}(u) - p_{i-1}(v)|^2}$$

Finally we compute the Eulerian integration:

$$v_i(u) = p_{i-1}(u) + v_i(u) \cdot dt$$

$$p_i(u) = c(p_{i-1}(u), p_{i-1}(u) + v_i(u) \cdot dt)$$

Where c is a constraint that fixes the position of the node to the original containing shere.

$$p(x, y) = \|x\| \cdot \frac{y}{\|y\|}$$

Until we reach convergence

$$\sum_{u \in H} \|p_i(u) - p_{i-1}(u)\|^2 < e$$

For some small value of e.

3.2. Node colouring

For node colouring we assign each color a value:

$$Pc(u) = Ac(u) * (1 - t) + Bc(u) * t$$

That is generated using the same exact strategy as $A(u)$ and $B(u)$, they are completely analogous since colour is also a 3D linear space, except that instead of using *randdir* we use an alternative procedure that generates a color in HSV space with random Hue and fixed Saturation and Value to 1.

3.3. Heuristic Rationale

In this subsection I will explain in detail the intention of each parameter and how it relates to the overall method.

3.3.1 A(u)

The rationale of the heuristic described in $A(u)$ can be difficult to grasp, my intuition was that we should place a node as close as possible to its less popular nodes ideally very close to the minium degree holder (of their $R(u)$).

The reason behind this idea is simple, there is low chance of finding a node that is connected to such niche node, if we by contrast decide not to place it near that node we are already loosing the possibility of creating any structure with that node, and the shape of the network will be rapidly decided by the greatest degree holders.

3.3.2 B(u)

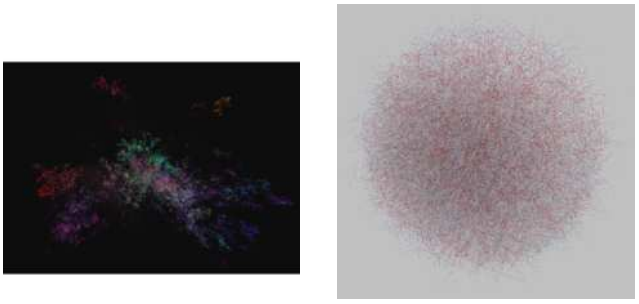
The rationale of the heuristic described in $B(u)$ is simpler, we want to separate all nodes as much as possible, the rule here was decided to be a random direction over a unit sphere multiplied by a constant.

This has the caveat that some nodes altough unrelated would have similar $B(u)$ positions, and in nodes where $B(u)$ dominates the final position (only the hubs) this can indeed be problematic. For this reason a fast optimization algorithm is proposed to precompute the positioning of $B(u)$ for the greatest degree holders taking into account heir degree.

3.3.3 C(u)

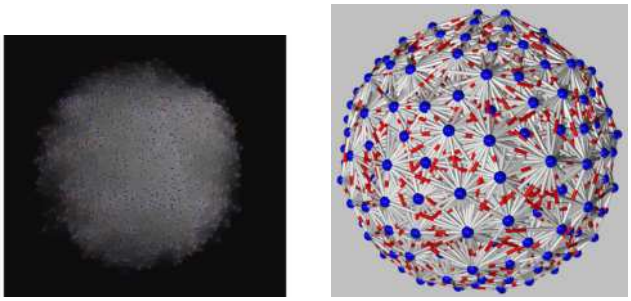
Finally the jitter function is included to prevent overdeterminism, for some nodes the positions of $B(u)$ and $C(u)$ will be the same when the node has the exact same adjacency list.

This is problematic because then the algorithm will collapse their position to the same spot in space, a bit of jitter can solve the problem and thus this situation will be represented in the visualization, if we have a lot of nodes with the exact



(a) Ours **(b)** Graphia

Figure 1: Barabási–Albert model comparison.



(a) Ours **(b)** Graphia

Figure 2: Erdős–Rényi model comparison.

same adjacency list they will be represented uniquely as an agglomeration of nodes.

3.3.4 Blending

Finally we blend parameters $B(u)$ and $A(u)$ using $l(u)$ this enables the soft transition between an exploration phase where we try to sparcify the network by separating the hubs as much as possible and an exploitation strategy where we try to place the nodes as close as possible to some other node.

Notice that we are implicitly taking into account the node degree of a node to sparcify it in case its a hub, this means that higher degree holders positions are naturally sorted, the higher ones will be farther from the center of the network than the smaller ones.

4. Results

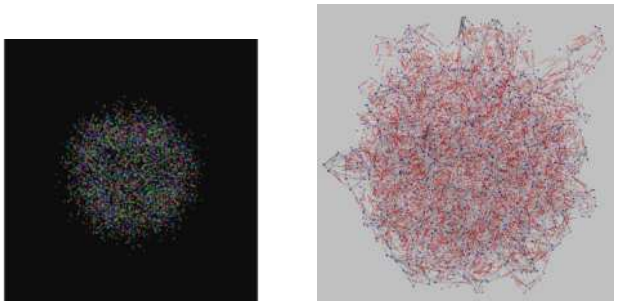
4.1. Synthetic model graph comparison

In this section I will show a comparison of the rendering of synthetic graphs between my layout algorithm and a stock one.

- Figure 1 shows Barabási–Albert with 1 million nodes, 5 initial nodes and 3 edges/node.
- Figure 2 shows Érdós–Renyi with 1000 nodes and connectivity probability of $p = 0.5$.
- Figure 3 shows Watts–Strogatz with 5000 nodes and rewiring probability of $\beta = 0.1$.
- Figure 4 shows Factor graph network with 150000 nodes.
- Figure 5 shows Randomized Factor graph network with 150000 nodes.

4.1.1 Effects on prefferential attachment power-law exponent

In this section I will present the results of applying the layout algorithm with $\tau - > 0.0, c = 0, b = 100$ to a prefferential attachment model with tunnable exponent (6).



(a) Ours **(b)** Graphia

Figure 3: Watts–Strogatz model comparison.

4.1.2 Effects on Watts–Strogatz rewiring probability

In this section I will shows how does the effect of the rewiring probability in the Watts–Strogatz model affect the layout algorithm (Figure 7, graphia vesion 8).

4.1.3 Erdős–Rényi variable connectivity probability

In this section I will shows how does the effect of the connectivity probability in the Erdós–Rényi model affect the layout algorithm in figure Figure 9.

4.2. Natural networks comparison

In this section I will show a comparison of the rendering of natural networks between my layout algorithm and a Graphia (Figures 10, 11, 12, 13).

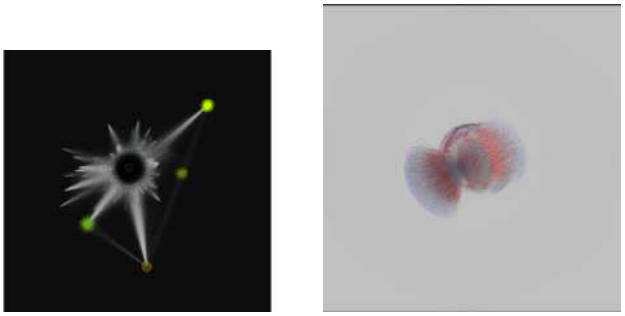
4.3. Layout parameter effects comparison

This section covers experiments on the effect of each parameter of the model independently, if let unspecified the parameters of the layout used are:

- b: 100.0
- c: 2.0
- τ: 4.0

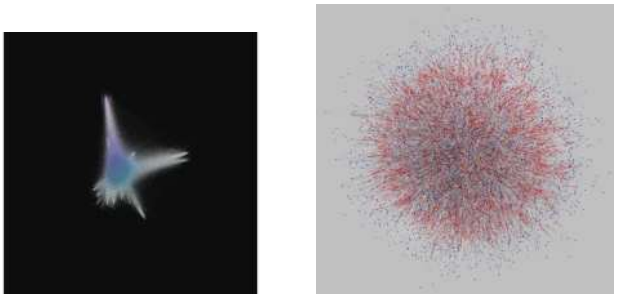
4.3.1 Parameter b effect

In this section I will show the effect of the parameter b of the layout algorithm with the Barabási–Albert network (Figure 14).



(a) Ours **(b)** Graphia

Figure 4: Factor graph model comparison.



(a) Ours **(b)** Graphia

Figure 5: Random factor model comparison.

The parameter b has a great effect in the overall scale of the network making it directly proportional, for visualization purposes scaling has been adjusted accordingly since we are interested in differences in the shape not in a constant scaling to every node.

4.3.2 Parameter c effect

In this section I will show the effect of the parameter c of the layout algorith with Barabási–Albert and Factor Graph network (Figure 15, 16).

4.3.3 Parameter τ effect

In this section I will show the main effect of the parameter τ, this parameter controls how biased the greedy strategy in the heuristic of $A(u)$ is (Figure 17).

4.4. Power-Law new model statistical measures

These results cover for both the non-randomized and randomized versions of the new Power-Law Model degree sequence presented earlier in log-log scale (Figure 18).

4.5. Rendering parameters effect comparison

In this section I will present the results of varying the rendering parameters of the visualization (Figures 19, 20). The parameters are highly dependent on the graph and the parameter of the layout algorith. What matters in this results is the effect they have in the rendering when modified, not their absolute values.

Also I will present how the colouring varies for different graphs, since there is only one colouring rule the results of the colourization are shown against a node degree heatmap colorization, meaningful that for each node in the network the control visualization is the same graph but colouring the nodes as a heatmap function over their node degree against the maximum node degree of the network (blue nodes indicate low degree and red nodes high degree).

5. Discussion

In this section I will cover the main findings in the experimentation.

5.1. Validity of the new power-law model

The Log–Log plots (Figures 18) of the degree rank vs node degree show a heavy tail behaviour but dont ultimately fit a power law in the strict sense, specially for the $\gamma = 2$ case.

What this plots show is a 2 regime function, it starts following a strict power-law function but at some point of the tail the node degree decreases with a much smaller rate than it should.

The theoretical estimation of a lower bound average for the node degree given its rank holds, however its proven to be quite a poor estimator (although not incorrect) for the second regime of the function.

There seems to be no significant difference in the node degree of both the deterministic and the randomized versions, which validates our initial hypothesis.

We can extract the following conclusions:

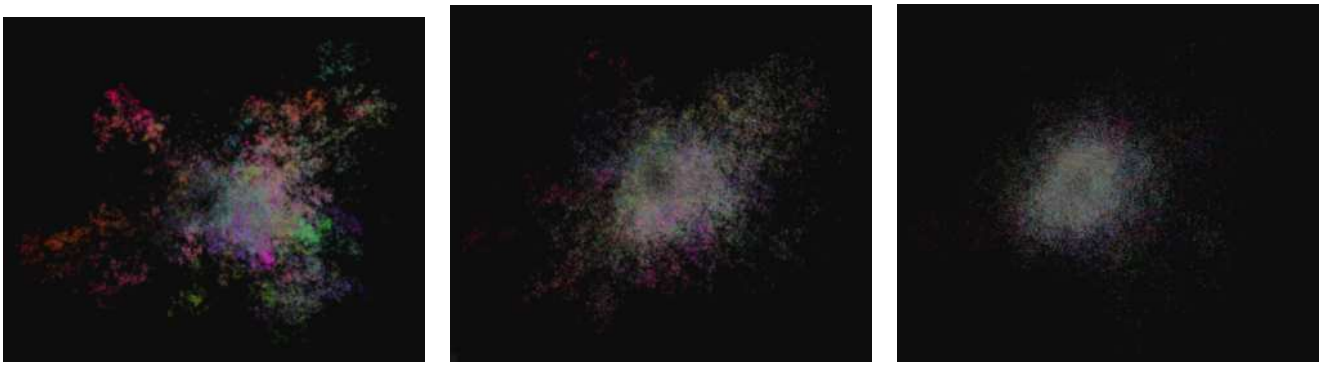
- The factor graph model shows a heavy tail behaviour characteristic of a Power-law model, but deviates rapidly from a standard Power-Law model when γ is increased beyond 1 showing a 2 regime model that is not accurately represented by my estimator of average node degree.
- The randomized factor graph model shows no significant statistical difference in node degre from the deterministic from a node degree perspective.

Although it does not follow strictly a Power-Law when the network is generated with $\gamma = 1$ it shows a similar heavy tail behaviour and given that real networks do not show a strict power-law degree sequence it still makes it a suitable network test our renderer against.

5.2. Comparison of the artifical models

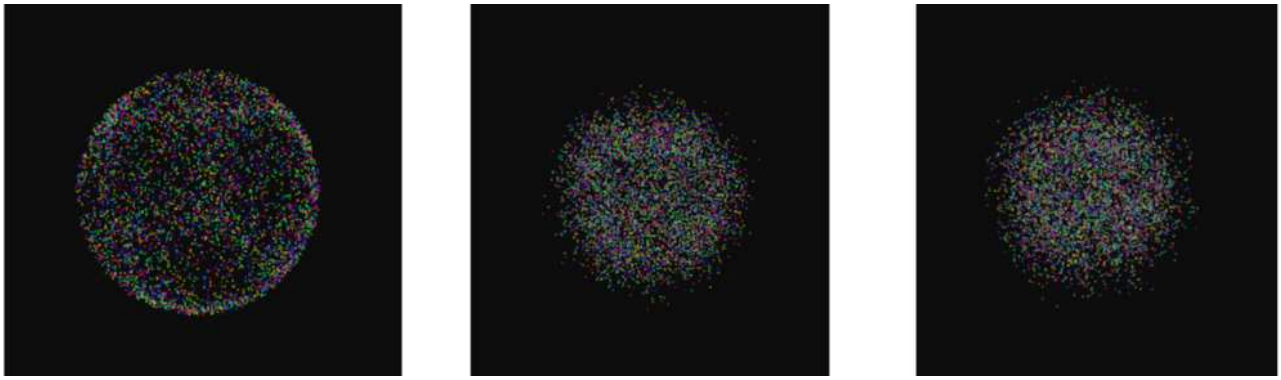
In this section I will discuss the comparison of my layout algoirithm against the stock one as seen in figure, first I will compare the results of the differnt models tested and then comparing them against the stock one. (Figures 1, 2 3, 4, 5)

The first interesting comparison comes from comparing Barabási–Albert with Erdós–Rényi models, we would expect a radically different behaviour in the layout of both given that the algorithm heuristic is thought only for the layout



(a) $\gamma = 2.82$ **(b)** $\gamma = 2.83$ **(c)** $\gamma = 2.86$

Figure 6: Barabási–Albert variable exponent layout



(a) $\beta = 0$ **(b)** $\beta = 0.1$ **(c)** $\beta = 1$

Figure 7: Watts–Strogatz network model on my renderer.

of Power-Law graphs. As we can see in the Erdós–Rényi model the layout algorithm places the nodes randomly in the surface of sphere without any structure, very different behaviour as opposed to Barabási–Albert which places them in a very structured way, where nodes with similar color are placed together and islands of nodes form. As opposed the stock algorithm does something similar to ourse with the Erdós–Rényi model but fails at giving any structure to the Barabási–Albert.

We can observe similar behaviour from the Watts–Strogatz model, where nodes are placed in the surface of a sphere randomly without any structure, however we can observe a small shift in the structure of the layout in the stock one as compared to the Erdós–Rényi model previous result.

5.2.1 Variation of γ exponent in Prefferential Attachment

The Figure 6 shows the variation of layout when rendering Barabási–Albert with a tunnable γ exponent.

The results show a fast and clear deviation from the standard model as we increase the value of γ. I have no clear explanation as to why there is such a sudden fall of perceived structure, as the exponent increases we obtain an unsaturated agglomeration of nodes in the middle without a prefferential direction. This requires further study.

5.2.2 Variation in rewiring of Watts–Strogatz model

In this section I will cover how does the rewiring probability affect to the layout of the algorithm. Interestingly (Figure 7) as the value β decreases towards 0 in our layout we can observe that points move from center towards the surface to making an almost perfect empty sphere.

In contrast, in the stock one (Figure 8) we can observe that nodes start to create a sense of structure until we reach a 2-arity graph giving a perfect loop.

This result is quite significant, it establishes that in graphs where all nodes have degree 2 our algorithm produces a sphere. We can reason why this happens easily.

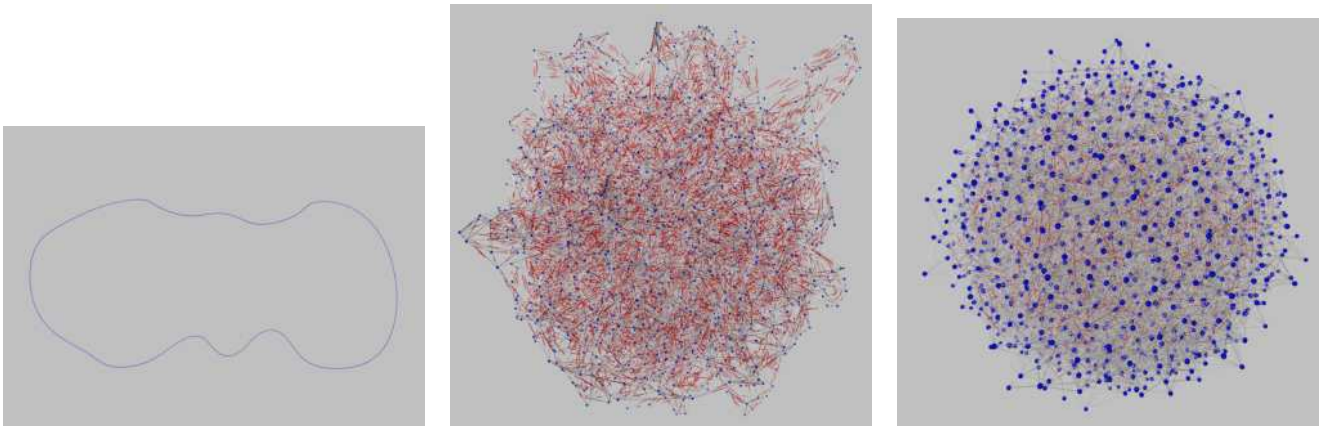
If all nodes share the same degree, then all nodes have the interpolation paramater equal to 1 since we have defined $t(u) = d_u/max(D(V))$, this means that $B(u)$ dominates in every node making the position of every node essentially:

$$P(u) = randdir() * b + randdir() * c$$

Which given that $b >> c$ this creates the exact expected result when $\beta = 0$ we have a rough surface sphere, all points are distributed equally on a sphere of $radius = b$ with an additional random jitter.

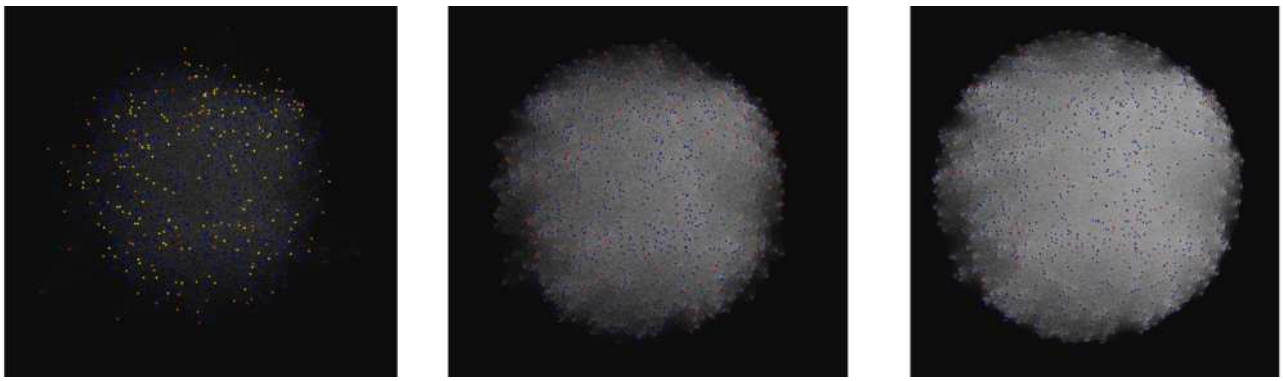
5.2.3 Variation of probability in Erdós–Rényi model

In this section I will treat how does the variaion of the probability in the Erdós–Rényi (Figure 9) model affect the render-



(a) $\beta = 0$ **(b)** $\beta = 0.1$ **(c)** $\beta = 1$

Figure 8: Watts–Strogatz network model on graphia renderer.



(a) $p = 0.088$ **(b)** $p = 0.5$ **(c)** $p = 0.957$

Figure 9: Erdős–Rényi network model.

ing.

As we can see from the representation (which has turned off the colouring of the nodes) decreasing the probability does not have the same effect that decreasing β in the Watts–Strogatz model had, for different ranges of values of probability we still roughly the same shape.

5.3. Comparison of the natural models

In this section I will analyse the comparison of the different natural networks tested with the stock one.

5.3.1 Email network

The email network (Figure 10) shows an interesting source of comparison, my algoirithm detects some nodes to be meaningfully connected with some high degree holders, creating small bubbles accros the space but ultimately placing most nodes in the middle, compared with the stock one which detects a simliar sense of structure but still connects them all with the middle. From interpretability point of view mine is capable of showing a more direct insight of this phenomena but the stock one gives a more balanced and clear look on the network.

5.3.2 Road network

The road network (Figure 11) which in my layout shows no clear structure or meaningful data. In this case the stock one gives a very differnt shape but still fails at showing a more meaningful insight about the network.

5.3.3 Google-WebGraph

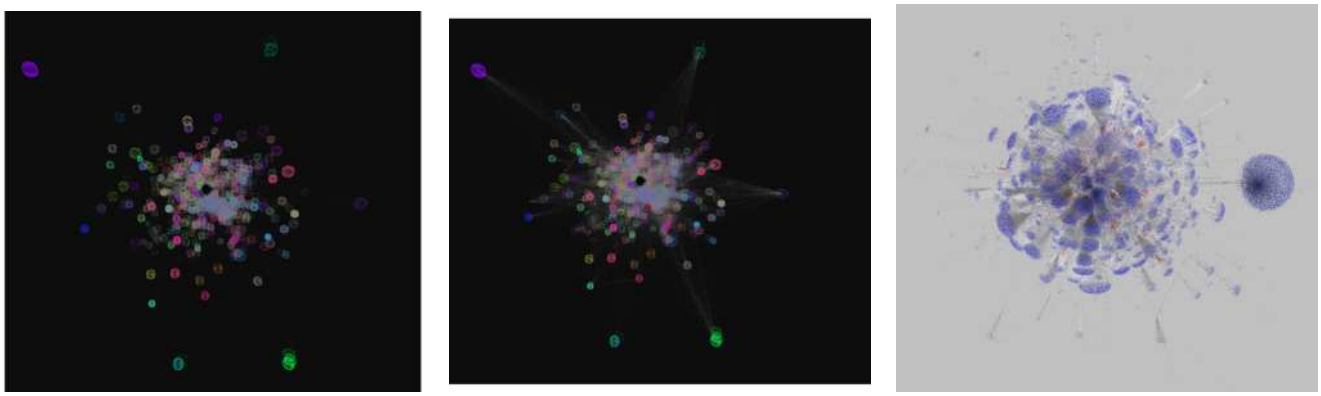
For the Google webgraph (Figure 12) my algorithm is capable of showing a sense of structure, identifying some high degree holders and placing them apart, however the interpretability of the network remains very poor. In the case of the stock one it fails largely at representating the WebGraph.

5.3.4 Linguistic Graph

For the linguistic (Figure 13) one we have a similar case to the email dataset, although this time the amount of ”bubbles” is much smaller, the same phenomena to a similar degree happens in the stock one.

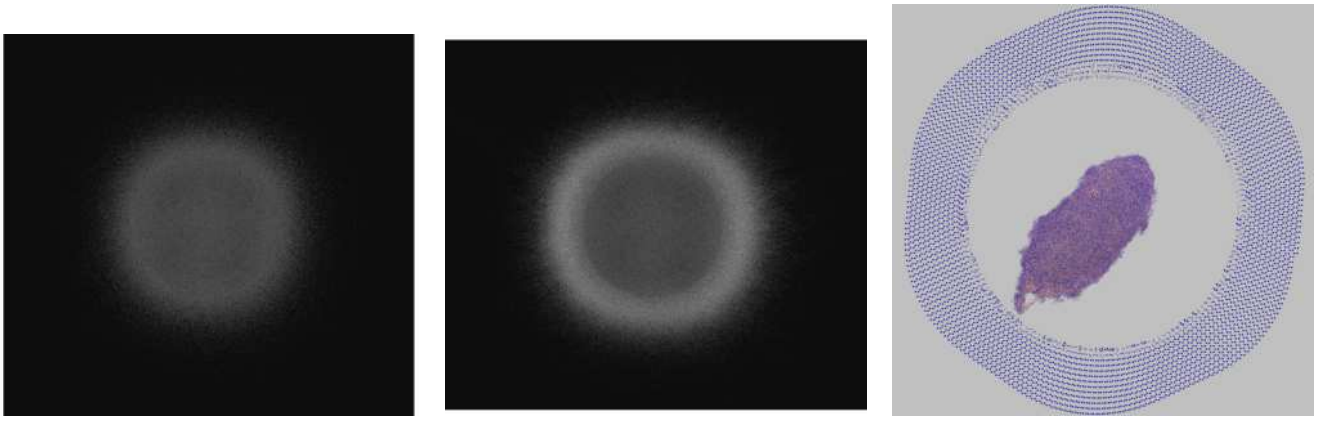
5.3.5 Summary

The results show that my layout algorithm is highly dependendant not only with the statistical properties of the node degree of the network but also with the structural nature of



(a) Email network **(b)** Email network edges **(c)** Email network (Graphia)

Figure 10: Email network



(a) Road network layout **(b)** Road network edges **(c)** Road network (Graphia)

Figure 11: Natural road network graph.

the graph, has great degree of veriation of results and does not serve as reliable tool to visulize natural graphs.

5.4. Effects of the layout parameters

In this section I will cover what are the main effects in the visualization of the multiple parameters of the algorithm.

5.4.1 Effect of the parameter b

As shown by Figure 14 the increase or decrease of parameter b mainly affects the overall scale of the network when rendered. But it can have an effect on small scales, what is relevant however is the ratio b/c.

This is not shown appropiately in the results but it can be easily reasoned, if $b- > 0$ then in our network every node u will have $B(u) = 0$, which means that all we are going to render the jitter noise that is combined from $A(u) + C(u)$. And essentially every node in the network regardless of the value of $A(u)$ will be randomly moved.

5.4.2 Effect of the parameter c

For the parameter c the behaviour observed in Figures 15 is the expected, when decreasing the parameter c the nodes of the network reduce its natural jitternes, reducing the component of randomization of the smaller degree nodes in the network. As we can see the overall network keeps its global structure and scale as parameter c increases to a certain degree, the visual result is a more blurry representation.

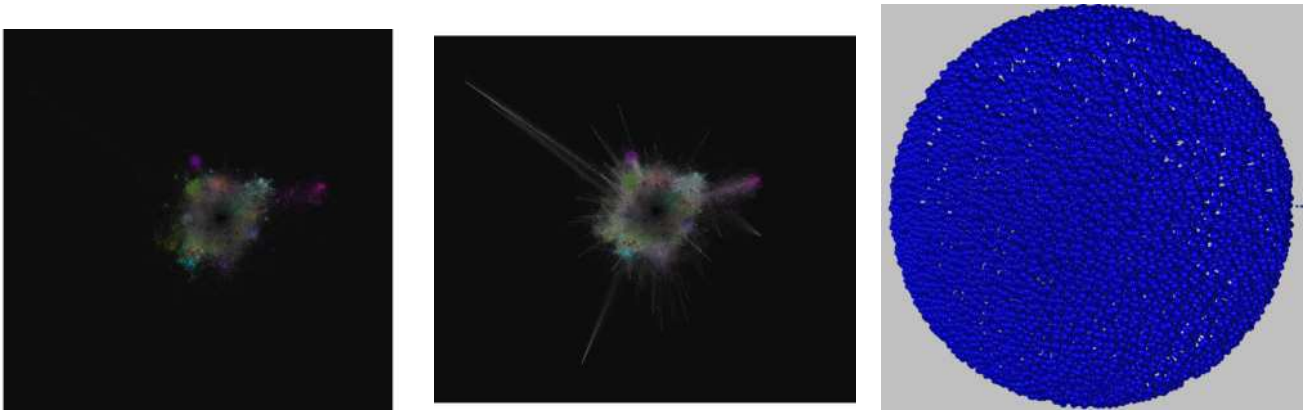
However this parameter is useful in case of nodes sharing the same adjacency list as shown in the Figure **??** of the factor graph network, this is the main cause of the observed ”bubbles” in the different visualization, therefore only make sense for gaphs where we expect to have nodes with an exact adjacency list. Because of how we have defined the factor graph above, we know there are substantial amount of nodes that share the same adjacency list, for instance every number that is a power of 2 is connected with 2, this gives a clear insight as to why this happens in this network and not in Barabási–Albert.

5.4.3 Effect of the parameter τ

For the parameter τ the behaviour observed in Figure 17 shows this is a key component of the ”fractalization” part of the visualization, and this have a particular effect in Barabási–Albert networks compared to other networks.

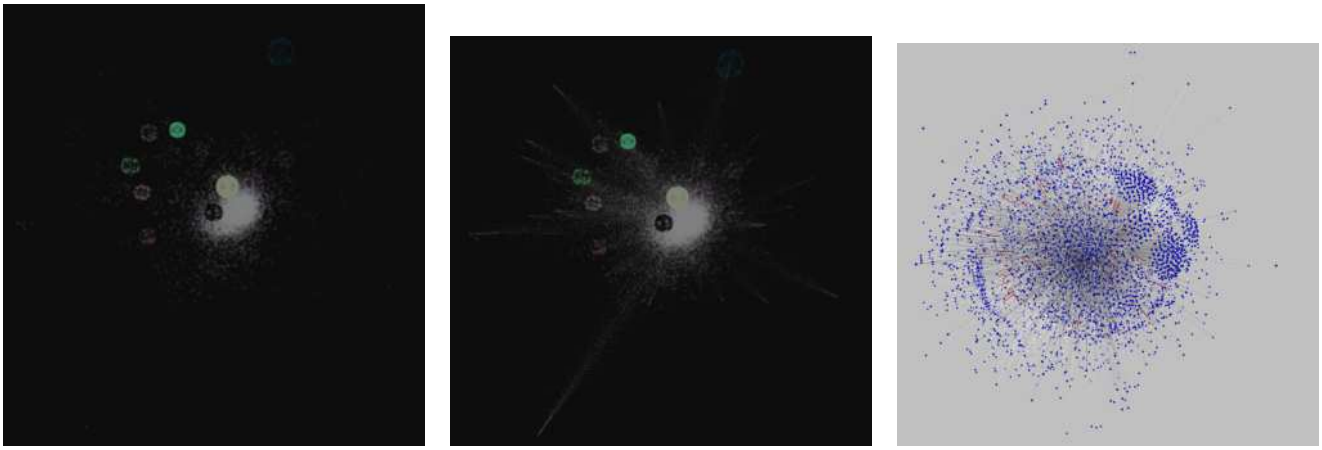
For the Barabási–Albert network this parameter clearly gives a hint of an underlying property of the network that gives this self similar structure. One way to measure how self similar a shape is to compute its fractal dimension.

Under traditional box counting algorithm, we count the num-



(a) Web graph layout **(b)** Web graph edges **(c)** Web graph (Graphia)

Figure 12: Natural web hyperlink graph.



(a) Catalan linguistic graph layout **(b)** Catalan linguistic graph edges **(c)** Catalan linguistic graph (Graphia)

Figure 13: Catalan linguistic dependency graph.

ber of bins that have been uniquely touched by the shape at multiple scales, then we compute the value D at each scale knowing that:

$$D = \lim_{\varepsilon \to 0} \frac{\log N(\varepsilon)}{\log\left(\frac{1}{\varepsilon}\right)}$$

And fitting a log log model, however I have not been able to measure this accurately, the landscape of point clouds makes the algorithm incorrectly report most network layouts as self similar. This been said, It's still left as open answer by this project why the Barabási–Albert model under my algorithm and with a $\tau = 0$ generates such very structured layout.

5.5. Effects of the rendering parameters

In this section I will cover some of the main effects of the rendering parameters (19, 20).

5.5.1 Effect of the colouring metric

The metric appropiately assigns different colours based on a random rule to the higher degree holders of the graph, intererestingly what is found across the different networks:

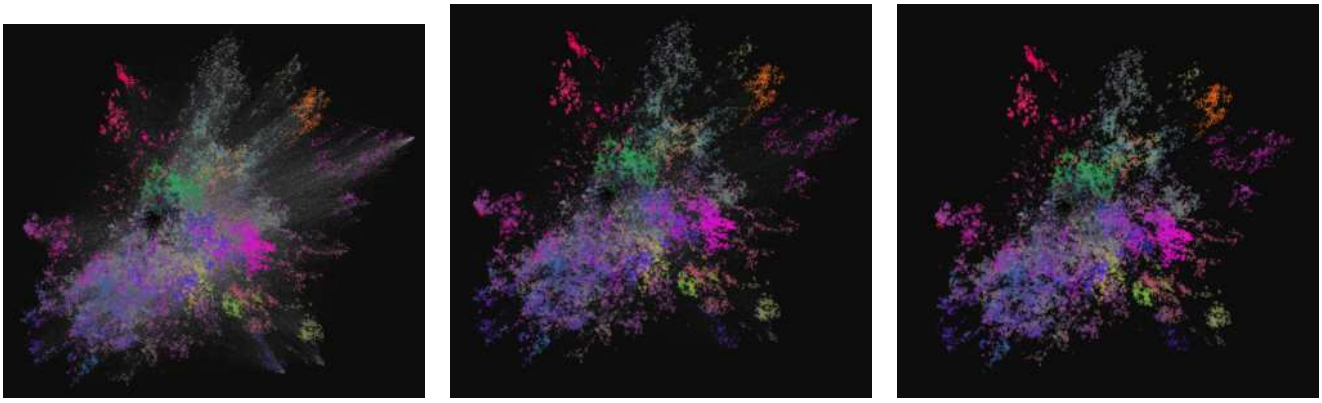
- Peers are slowly contaminated in colour through their neighbours, meaning that only closely related peers (those that have few connections and mainly to the higher degrees) have the most saturation amongst the low degree nodes.

- Nodes that belong to a "bubble" (As we have established nodes that are only connected to one node) inherit their colour directly.

- Colouring inherits the same problematic as the assigment of $B(u)$ from a probabilistic point of view big degree holder nodes can be assigned a very similar colour, a better heuristic is necessary to optimize the seperation of the colouring.

Finally, the colour can be interpreted as the "purity" of a node. That is in broad terms, the more connected a node is, specially to already less pures nodes the less saturated its colour is on average and closer it approaches to gray, this can be useful to identify parts of the graph that are closely connected to a particular high degree holder instead to the rest of the network.

Further study is necessary to precise the notion of "purity" and how different networks topologies show the variations of colours beyond what has been studied here.



(a) $b = 2$ **(b)** $b = 10$ **(c)** $b = 100$

Figure 14: variabiltiy of parameter b with Barabási–Albert network model of 50000 nodes $b \in (2, 10, 100)$

(a) $c = 0$ **(b)** $c = 1.358$ **(c)** $c = 4.482$

Figure 15: variabiltiy of parameter c with Barabási–Albert network model $c \in (0, 1.358, 4.482)$

5.5.2 Effect of alpha blending and line thickness

Alpha blending has been proven to be effective way to reduce graph messyness when multiple nodes are agglomerated together, and gives a clearer inside especially in the density of nodes in particular region, as opposed to traditional argument that apply a binary colouring (either there is node or there isn't) we can directly visualize the density of nodes in a given region, same goes for edges.

Further study is necessary to for instance, compute for every network what is the most adequate value of alpha blending and line thickness so we can observe connectionk and layout density accurately without loosing information.

6. Conclusions

This section will give a clear answer to every *Research Question* presented in the *Introduction* of the article as well as a final conclusion.

6.1. RQ1

Most of the times it does not, only the WebGraph, Emails, and Linguistic graphs show some degree of interpretability, while roads is completely unintrepretable.

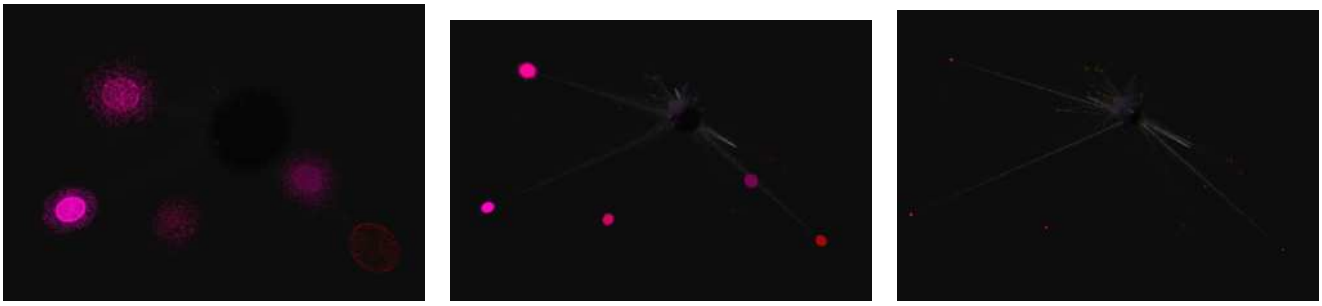
6.2. RQ2

The usefulness of the layout algorithm if any is very limited, while it creates interesting forms in the Barabási–Albert and Watts–Strogatz networks, has this concept of "Bubbles" (collection of nodes that are connected to only one node), and provides a unique way to visually see what is the connectivity relationship between nodes it is still quite lacking on the natural ones.

Its only usefulness is it in the study of networks that follow simialr behaviour to traditional Barabási–Albert which is not very abundant, and still the interesting shape created by my algorithm of such network can be seen more as an interestign artifact not something meaningful. However further study is necessary on why this happens and how does it relate to the model, and wht other networks do not show same behaviour.

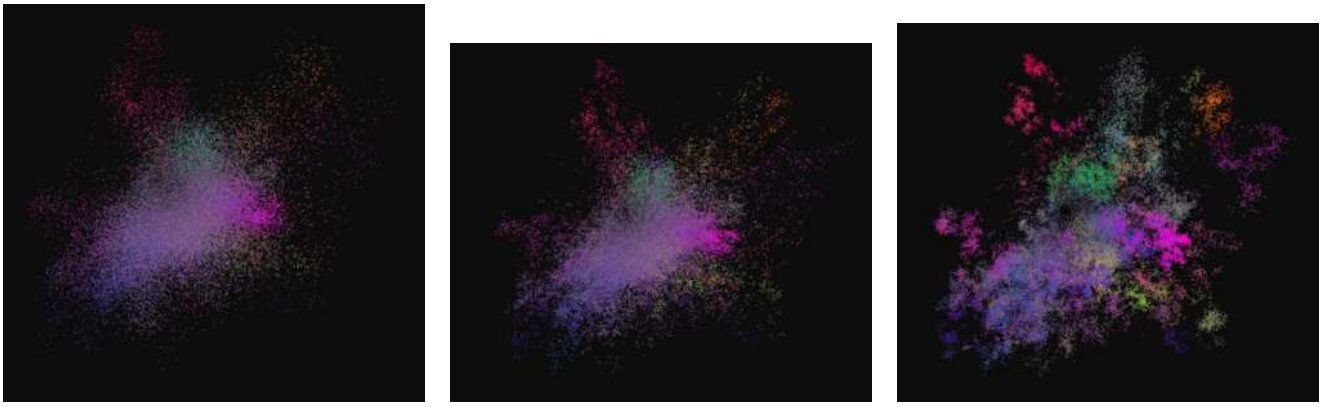
6.3. RQ3

The behaviour of other types of graphs that do not follow a power-law degree sequence (only tested for binmomial) has shown to be interesting, depending on the variabiltiy of the average degree of the network it creates spheres of different thickness, at least for the synthetic models tested.



(a) $c = 6.218$ **(b)** $c = 1.094$ **(c)** $c = 0$

Figure 16: variabiltiy of parameter c FactorGraph with 150000 nodes $c \in (0, 1.094, 6.218)$



(a) $\tau = 4$ **(b)** $\tau = 1$ **(c)** $\tau = 0$

Figure 17: Effect of evolution of parameter τ on Barabási–Albert networks with 50000 nodes.

6.4. RQ4

Considering only power-law degree sequence graphs, improvement is minimal, only for the tested case of WebGraph and Barabási–Alber the algorithm seems to give a better layout, still offers and interesting line of study for future versions of this algorithm, there are many open questions that this article has left on the behaviour of this algorithm on some networks, its not totally clear and there are interesting structures left unexplained.

6.5. RQ5

The parameter that has been found to be less relevant is τ which for the most graphs to give a pleasant easily interpretable figure it has to be set almost to 0, switching the softmin function to just a function that picks the lowest degree (or lowests degree if there is any draw) and performs the average.

The parameter c only has taken part when the network presents "bubbles", if none a smaller value, ideally 0 gives better interpretation of the network, specially for very large graphs and Barabási–Albert.

6.6. Final conclusion

This model while visually pleasing and providing a fast layout heuristic fails to efficiently layout and render large power-law graphs. Its not superior to already established algorithms in quality terms alone and its use its not recommended. However it produces interesting results for Barabási–Albert and Factor Graph but that is simply a visual curiosity than a relevant matter in graph plotting.

7. Methods

In this section I will cover the methodology used to conduct the experiments and justify its correctness.

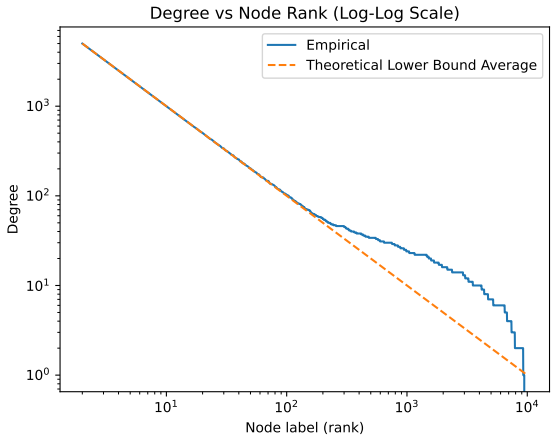
7.1. Choice of parameters and experiments

One of the main difficulties of experimenting with this project is the high number amount of parameters and models, it would be very time consuming to run one experiment per every pair of possible set of layout parameters and network models for a varying degree of them. For the comparison of the layout algorithm parameters with Graphia the following parameters have been used:

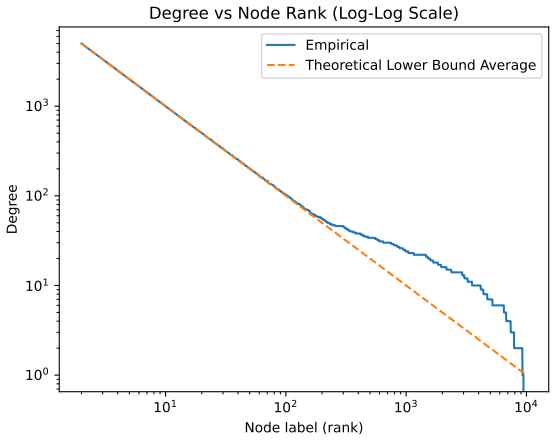
Network	**c**	**b**	τ
email-EuAll	4	100	0*
roadNet-CA	0	100	0*
Linguistic Catalan	0	100	0*
Google-WebGraph	0	100	0*

Table 2: Parameters used for the layout algorithm across the different networks: c, b, and τ. *0 means limit to 0 but not 0.

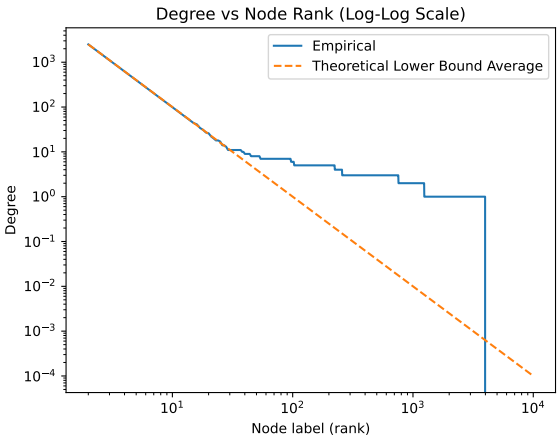
Since the comparison was about giving the clearest possible layout for each given network and not assesing the influence



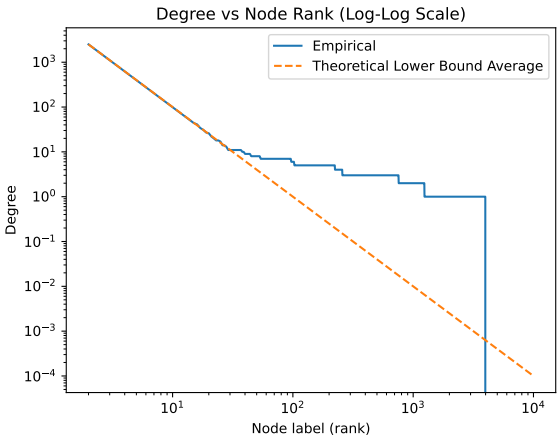
(a) With hashing $N = 10{,}000$ and $\gamma = 1$



(b) Without hashing $N = 10{,}000$ and $\gamma = 1$



(c) With hashing $N = 10{,}000$ and $\gamma = 2$



(d) Without hashing $N = 10{,}000$ and $\gamma = 2$

Figure 18: Log–log plots of node degree versus rank for $N = 10{,}000$ and $\gamma = 1$, comparing randomized (hashed) and deterministic labelings across two independent realizations.

of any parameter.

7.2. Synthetic networks

For the synthetic graphs used the default parameter of the generators are (if let some parameter unspecified, the default was used):

- **Erdős–Rényi (ER)**: $N = 100$, $p = 0.05$
- **Barabási–Albert (BA)**: $N = 100$, initial nodes $m_0 = 5$, edges per new node $m = 3$
- **Generalized Preferential Attachment**: $N = 100$, $m_0 = 5$, $m = 3$, $\gamma = 3.0$, $k_0 = 1.0$
- **Watts–Strogatz (WS)**: $N = 100$, nearest neighbors $k = 4$, rewiring probability $\beta = 0.1$
- **Factor Graph Fast**: $N = 100$, $\gamma = 1$
- **Factor Graph Fast Seed**: $N = 100$, $\gamma = 1$, seed = 1.0

7.3. Data Validation

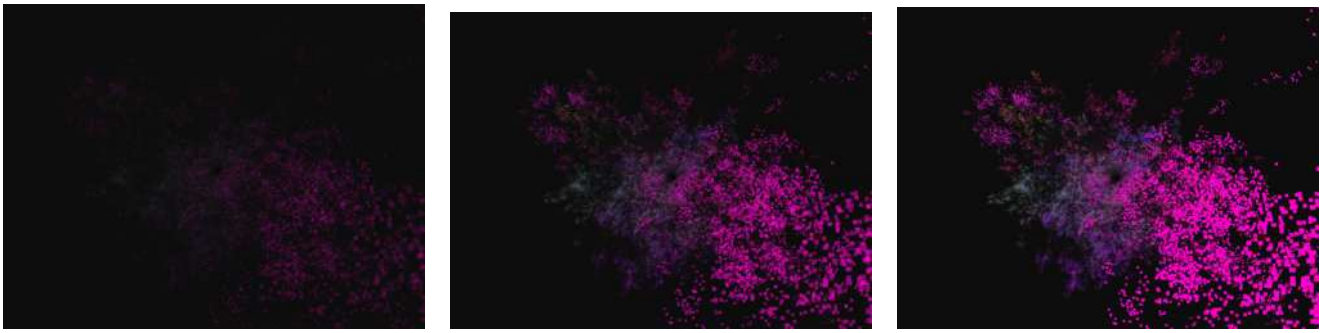
The visualization program counts with a varying degree of metrics to validate the data, first of all it display the number of nodes and edges as well as well as node degree plots in both linear and log scale. Then during the visualization the degree of the nodes can be easily tested visually using the debug options, this helps verify the assumption and placement of the nodes and correctly check that indeed the algorithm is placing the higher degree nodes following the described algorithm.

7.4. Power-law degree sequence plot

Given the fact that the model follows exactly the power-law distribution on the given lower bound formula for its expected degree it has not been tried to fit a Power-Law model, mainly for these reasons:

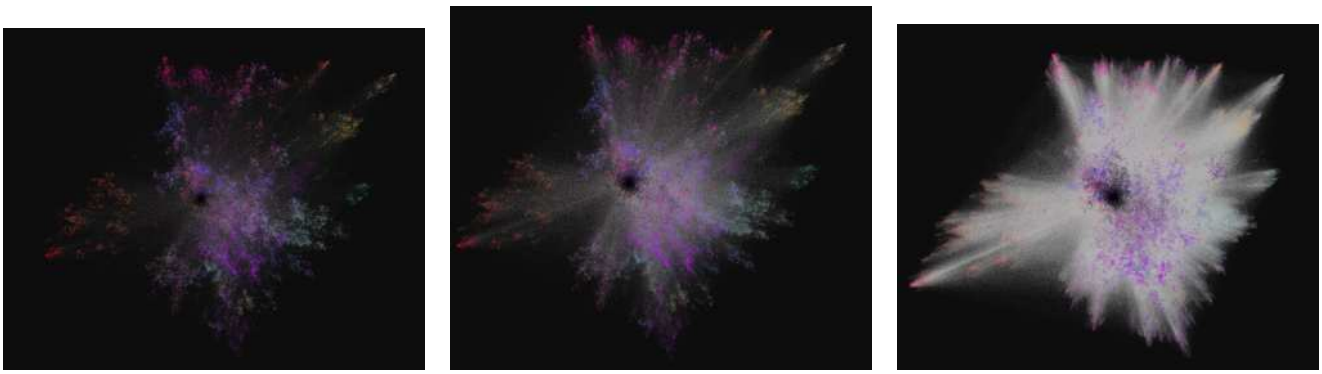
- What the data shows is that the model follows strictly a power-law distribution until a breaking point.
- From that point the model deviates from a traditional power-law distribution, with a higher significant degree as γ increases.

These removes the need to fit a regular Power-Law distribution since its clear from the data that this is not the actual underlying distribution, however its explained its chosen



(a) $alpha = 0.11$ **(b)** $alpha = 0.50$ **(c)** $alpha = 1.0$

Figure 19: Effect of alpha / transparency parameter on rendering.



(a) $EdgeThickness 0.000018$ **(b)** $EdgeThickness = 0.000064$ **(c)** $EdgeThickness = 0.000598$

Figure 20: Effect of number of edges on rendering.

because when $\gamma = 1$ we can observe a similar behaviour and the effects of the second regime are not are not very exaggerated. Even if we succeeded at fitting a Power-Law model like we had in some labs of the subject it would be inherently incorrect.

7.5. Implementation Note

The project experiments have been executed under self coded visualization tool hosted in gtihub.com under repositttory *detravisualizer* David Nexuss (2026), it's a C++ CMake project working with OpenGL to perform the visualization.

The program can load graphs in format of pairs of edges (source, target) in regular plain text, from a compressed Gzip text file (typical format of SNAP database graphs) and regular conllu files (for which it creates the aggregated graph automatically).

Some videos and images are provided under subdirectory showcase of the github repository containing extra visualizations of the algorithm.

References

David Nexuss (2026). Detravisualizer: Graph visualization tool. `https://github.com/DavidNexuss/detravisualizer`.

Graphia Developers (2024). Graphia: Open-source graph visualization software for big data. `https://graphia.app/`.

Nivre, J. and et al. (2016). Universal dependencies v1: A multilingual treebank collection. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC)*, Portorož, Slovenia.

Stanford Network Analysis Project (2016). Stanford large network dataset collection. `https://snap.stanford.edu/data/`.

Taulé, M., Martí, M. A., and Recasens, M. (2008). Ancora: Multilevel annotated corpora for catalan and spanish. In *Proceedings of the 6th International Conference on Language Resources and Evaluation (LREC)*, Marrakesh, Morocco. Universal Dependencies Catalan AnCora treebank used; converted from AnCora original corpus.