Benchmarking the performance of ForceAtlas2 and other algorithms on Large Networks

Social Network Analysis for Computer Scientists — Course paper

Ernest Vanmosuinck
s3210359@umail.leidenuniv.nl
LIACS, Leiden University
Leiden, Netherlands

Mouni Priyanka Thandu
s3420361@umail.leidenuniv.nl
LIACS, Leiden University
Leiden, Netherlands

ABSTRACT

With the ever-expanding growth of social network analysis, the development of efficient, accurate and scalable visualization algorithms is a fascinating topic. In this research, we expand Jacomy et al's research [11] by benchmarking a total of five different visualization algorithms; ForceAtlas2, ForceAtlas2 LinLog, OpenOrd, Yifan Hu and Yifan Hu Proportional, all available on the Gephi framework, on a set of new larger datasets, using the inverted Noack's normalized^{endv} atedge measure. We find that the performance of ForceAtlas2 is not as prominent when evaluating larger networks, and emit doubt as to the usability of Noack's measure to correctly benchmark visualization algorithms.

KEYWORDS

Visualization Algorithms, Gephi, ForceAtlas2, Yifan Hu, OpenOrd, benchmarking, social network analysis, network science

ACM Reference Format:
Ernest Vanmosuinck and Mouni Priyanka Thandu. 2022. Benchmarking the performance of ForceAtlas2 and other algorithms on Large Networks: Social Network Analysis for Computer Scientists — Course paper. In *Proceedings of Social Network Analysis for Computer Scientists Course 2022 (SNACS '22).* ACM, New York, NY, USA, 8 pages.

1 INTRODUCTION

In this paper, we aim to study the continuous graph visualization algorithm called ForceAtlas2, and analyse its performance on a variety of datasets, borrowing from the benchmark suite approached by Jacomy et al [11]. We also wish to expand the benchmark performed to include new algorithms; Yifan Hu Proportional and OpenOrd [30]. Comparing with OpenOrd will also enable us to evaluate the benchmarking measure used in the baseline paper on a different type of algorithm.

2 PROBLEM STATEMENT

Gephi [2] is a network visualization program that includes network spatialization as one of its core features, and one of its main visualization algorithms is ForceAtlas2. ForceAtlas2 has been developed as a comprehensive solution for visualizing networks. Jacomy et al [11] claim to not have made a theoretical breakthrough, but rather an attempt to integrate many methodologies together such as the Barnes Hut simulation, degree-dependent repulsive force, and local and global adaptive temperatures. In this paper, we compare the performance of ForceAtlas2 which runs in a continuous homogeneously layout with other algorithms and expand the research on larger real-world networks.

SNACS '22, Master CS, Fall 2022, Leiden, the Netherlands

3 RELATED WORK

Our research is using the work presented by Jacomy et al. [11] as baseline for this paper. In their research, they compared their implementation to three different algorithms; the LinLog variant of ForceAtlas, the Yifan Hu algorithm [10] and the Fruchterman-Reingold algorithm [7], showing that the improvements and performance of ForceAtlas2 were quite noticeable over the other algorithms already available on Gephi.
The ForceAtlas2 algorithm has since been used in loads of further research for the visualization of complex networks. It is concentrated on accuracy and being helpful for exploring real data, allowing for a rigorous interpretation of the graph, for example in Social Network Analysis, with the fewest biases possible, and decent readability even if it is slow. Fruchterman-Reingold algorithm [7] mimics the graph as a mass particle system. The nodes represent the mass particles, while the edges represent the connections between the particles. The algorithm attempts to reduce energy consumption. It has become an industry standard, although it is still incredibly slow.
We also acknowledged the existence of the GraphViz framework [5], an open-source software developed in C that is also capable of visualizing large networks. While faster due to its development in the C language, the incorporation of this tool is out of scope for this project, and will thus not be explored.

4 ALGORITHMS

4.1 ForceAtlas2

ForceAtlas2 is a force-directed architecture that spatializes a network. While edges pull their nodes in like springs, nodes repel one another like charged particles. These forces produce a movement that eventually reaches equilibrium. The interpretation of the data is anticipated to benefit from this final configuration. Each node is specifically positioned in the force-directed drawing in relation to the other nodes. Only the connections between the nodes determine how this process works. Nodes' eventual properties are never taken into consideration. There are problems with this approach. Depending on the initial state, the outcome differs. The procedure

could become stalled at a local minimum. The coordinates of each point do not correspond to any particular variable, and it is non-deterministic. It is impossible to read the outcome as a Cartesian projection. A node's location cannot be interpreted on its own; instead, it must be compared to those of the other nodes. Despite these drawbacks, one benefit of the approach is that it makes the structure visible. Its fundamental function is to convert structural proximities into visual proximities, which makes it easier to analyze data, particularly data from social networks.

4.1.1 *Energy Model*

The attraction force and the repulsion force are the foundation of every force-directed algorithm. The 'spring-electric' [4] layout is inspired from real life and uses repulsion formula of electrically charged particles ($F_r = k/d^2$) and the attraction formula of springs ($F_a = -k.d$) where d is the geometric distance between two nodes. Going forward, Fruchterman and Rheingold [6] created an efficient algoritm using custom forces: repulsion force ($F_r = -k^2/d$) and attraction force ($F_a = d^2/k$) with k adjusting the scaling of the network.

The author Noack in this paper on Energy models for clustering [32] explains the importance of distance in the spatialization of graphs as the key distinction between force-directed algorithms. The strength of the forces in physical systems depends on the proximity of the interacting components: closer entities repel more strongly than more distant ones and vice versa. Forces and distance can interact in a linear, exponential, or logarithmic fashion. Noack defines a layout's energy model or "(attraction, repulsion)-model" as the exponent multiplied by the distance in the equations used to compute attraction and repulsion.

For example, as mentioned by Jacomy et al. [11], the model of the spring-electric layout is (1, -2). The (attraction, repulsion)-model of ForceAtlas (1, -1) falls somewhere between Noack's LinLog (0, -1) and Fruchterman and Rheingold's method (2,-1).

Noack [34] states that "distances are less dependent on densities for large, and less dependent on path lengths for small ". The ratio of actual edges to prospective edges is the "density." It indicates that visual clusters signify structural densities when is low, i.e. when the attraction force is less dependent on distance and the repulsion force is more dependent on it. ForceAtlas2's ability to display clusters outperforms Fruchterman and Rheingold's method but falls short of LinLog.

A classical attraction force:

The attraction force F_a. between two connected nodes n_1 and n_2 is insignificant. It depends linearly on the distance d(n_1, n_2). This can be represented as:

$$F_a(n_1, n_2) = d(n_1, n_2) \tag{1}$$

Repulsion by degree:

The goal is to bring weakly connected nodes closer together with well-connected ones. Our idea is to reduce the repulsive effect between a highly linked node and a weakly connected node. As a result, they will be closer to a balanced condition. The repulsion force F_r is proportional to the product of the degrees plus one (deg + 1) of the two nodes, n_1 and n_2.

The formula is represented as:

$$F_r(n_1, n_2) = k_r \frac{(deg(n_1) + 1)(deg(n_2) + 1)}{d(n_1, n_2)} \tag{2}$$

The equation is similar to what Noack proposes in the paper [32], with only only one change of (deg+1) instead of deg. This is significant since it assures that even nodes with a degree of 0 have some repulsion force. The authors believe that this feature has a greater influence on the outcome and readability than the (attraction, repulsion)-model.

4.1.2 *Settings*

In this section, the paper discusses the parameters, modes and their effects. We now go through the options shown to the user, what they do, and how they affect the layout during Gephi implementation. The majority of these options allow the user to influence node location and sometimes the shape of the network. They may be enabled while the layout is running, enabling the user to observe how they affect spatialization.

- **Linlog mode:** The LinLog mode uses a logarithmic attraction force with the following formula:

  $$F_a(n_1, n_2) = log(1 + d(n_1, n_2)) \tag{3}$$

  The formula is different from Noack's [33] because of the added value 1 to the distance. When switching from normal to LinLog mode, the scale value must be re-adjusted. The energy model has a strong impact on the shape of the graph, and the time of convergence.
- **Gravity:** Gravity is a popular enhancement of force-directed designs. This force prevents unconnected nodes from dispersing. It draws nodes to the spatialization space's centre. Its primary function is to compensate for repulsion for nodes located far from the centre. In our situation, it must be weighted similarly to repulsion. Force of gravity is represented as:

  $$F_g(n) = k_g(deg(n) + 1) \tag{4}$$

  where k_g is set by the user. There is also a strong gravity option that attracts nodes that are far distant from the centre (this is the distance d(n)), but this force is so strong that it has its drawbacks. It sometimes might lead to biased placement of the nodes in the graph. However, its advantage is that the output graph will be more compact.

  $$\dot{F_g}(n) = k_g(deg(n) + 1)d(n) \tag{5}$$
- **Scaling:** We have two constants, attraction constant k_a and repulsion constant k_r which adjust the attraction and repulsion forces respectively. These two constants play opposite roles. Increasing the attraction constant k_a reduces the size of the graph whereas increasing the size of repulsion constant k_r expands the graph. However, in practical use, it is preferable to have only one scaling parameter. In ForceAtlas2 there is no attraction constant k_a. We use k_r to scale the size of the graph as required.
- **Edge weight:** In the case of weighted edges, the weight will be taken into account when calculating the attraction force. δ is used to represent 'Edge weight influence. If δ is set to 0,

the weights are ignored, and if δ is set to 1, the attraction is proportional to the weight. Any values set above 1 emphasize the weight effects.
- **Prevent Overlapping:** The repulsion is modified in this mode such that the nodes do not overlap. The idea is to create a more readable and visually appealing image. The idea is to take the size of the nodes n_1 and n_2 into account while computing the distance d(n_1,n_2) for both in repulsion force and attraction force.
  It is observed that when this mode is switched on the convergence isn't as smooth and the spatialization is considerably slowed. It is important to keep in mind to apply this mode after the convergence.

4.2 Yifan Hu

The Yifan Hu layout algorithm [10] combines the best features of force-directed algorithms to minimize algorithm complexity. This is one of the algorithms that perform admirably in large networks. It is an extremely fast method that performs well on huge networks. To minimize complexity, it combines a force-directed model with a graph coarsening approach. A Barnes-Hut [1] computation, which considers them as one super-node, approximates the repulsive forces on one node from a cluster of distant nodes. The most prominent parameters for the Yifan Hu algorithm include:

- **Step ratio:** A high ratio boosts quality at the expense of speed
- **Theta:** A smaller Theta results in more accurate results but takes more time
- **Optimal distance:** Controls distance between nodes

4.3 Yifan Hu Proportional

The Yifan Hu Proportional layout algorithm is identical to the Yifan Hu layout algorithm, except that it utilizes a proportional displacement strategy for node placement in the graphical space instead of a proportional displacement strategy. The accuracy and speed are virtually identical to the Yifan Hu algorithm.

4.4 OpenOrd

The OpenOrd algorithm [30] is one of the few force-directed layout algorithms that can expand to more than one million nodes, making it suitable for big networks. According to the authors of [30], it is very quick, scales to millions of nodes can be run on multicore processors, and aims to emphasize clusters. It anticipates undirected weighted graphs and strives to distinguish clusters more effectively. It may be run in parallel to speed up computation and automatically shuts down. The method is based on Frutcherman-Reingold [7] and operates with a predetermined number of regulated iterations. To allow clusters to split, long edges are clipped. The most prominent parameters for the OpenOrd algorithm include:

- **Edge Cut:** Higher values show more clustered results in the networks
- **Num Iterations:** Higher values show more expanded results in the networks

5 DATA

To perform a proper analysis of our implementation, we aim to use the dataset used by Jacomy et al. [11]. They employed a total of 68 different datasets, most of which come from the Stanford Large Network Dataset Collection (referred to as SLND) [27]. We provide a small definition of all the networks that they used;

- **Facebook**: 10 ego networks consisting of friends list from Facebook, anonymized to maintain user safety [23, 31].
- **Twitter**: while the Twitter dataset from SLND contains a total of 973 networks [24, 31], Jacomy et al. [11] only used 30 of those networks. They extracted the 10 "largest" networks, the 10 "smallest" networks, and 10 "medium" sized networks, the size criteria determined by the byte size of said network.
- **Oregon-2**: 9 autonomous systems graph, representing AS peering information inferred from Oregon route-views [12, 25].
- **COND-MAT**: a collaboration network covering scientific collaboration between authors' papers submitted to the Condense Matter category on arXiv [13, 26], covering the period from January 1993 to April 2003. If an author *i* co-authored a paper with author *j*, the graph contains an edge from *i* to *j*. If the paper is co-authored by *k* authors this generates a completely connected (sub)graph on *k* nodes.
- **GR-QC**: a collaboration network covering scientific collaboration between authors' papers submitted to the General Relativity and Quantum Cosmology category on arXiv [16, 26], covering the period from January 1993 to April 2003. If an author *i* co-authored a paper with author *j*, the graph contains an edge from *i* to *j*. If the paper is co-authored by *k* authors this generates a completely connected (sub)graph on *k* nodes.
- **HEP-PH**: a collaboration network covering scientific collaboration between authors' papers submitted to the High Energy Physics - Phenomenology category on arXiv [19, 26], covering the period from January 1993 to April 2003. If an author *i* co-authored a paper with author *j*, the graph contains an edge from *i* to *j*. If the paper is co-authored by *k* authors this generates a completely connected (sub)graph on *k* nodes.
- **Karate**: a social network of friendships between 34 members of a university karate club [29, 39].
- **Heroes** Social Network: a network of Marvel superheroes, collected by Cesc Rosselló, Ricardo Alberich, and Joe Miro from the University of the Balearic Islands. The dataset was then transformed and enhanced by Kai Chang.
- **C-Elegans**: a dataset representing the neural network of C. Elegans [29, 38]. Note that this is a different dataset than the C. Elegans dataset proposed by the SLDN, as the latter does not have the same number of nodes and edges. The dataset used in this paper is available through the Gephi dataset repository, as are the Karate, Heroes and Yeast datasets [29].
- **Yeast**: a network consisting of protein-protein interaction in yeast [3].

Additionally, we want to test the implementation on larger networks, as the biggest network presented by Jacomy et al. [11] was a

network of 23,133 nodes and 186,936 edges. We extend this research with the following datasets of larger networks:

- **cit-HepPh**: a citation graph covering all the citation within a dataset of 34,546 papers on high energy physics phenomenology submitted on arXiv [8, 20, 25]. If a paper *i* cites paper *j*, the graph contains a directed edge from *i* to *j*. It does not include citations outside the dataset.
- **cit-HepTh**: a citation graph covering all the citations within a dataset of 34,546 papers on high energy physics theory submitted on arXiv [8, 21, 25]. If a paper *i* cites paper *j*, the graph contains a directed edge from *i* to *j*. It does not include citations outside the dataset.
- **Enron Email**: a network covering all the email communications within a dataset of around half a million emails [14, 28]. Nodes of the network are email addresses and if an address *i* sent at least one email to address *j*, the graph contains an edge from *i* to *j*.
- **Facebook page-page**: represent a page-page network of verified Facebook sites [15, 36]. Nodes represent Facebook pages while edges represent mutual links between sites.
- **Gnutella24**: sequences of snapshots from the Gnutella peer-to-peer file sharing system taken from August 24th, 2002 [17, 26].
- **Gnutella25**: sequences of snapshots from the Gnutella peer-to-peer file sharing system taken from August 25th, 2002 [18, 26].
- **math-overflow**: four networks covering interactions on the stack exchange website Math Overflow [22, 35]. The temporal element is ignored. There are three different types of interactions represented by a directed edge (u, v):
  - user u answered user v's question (in the graph sx-mathoverflow-a2q)
  - user u commented on user v's question (in the graph sx-mathoverflow-c2q)
  - user u commented on user v's answer (in the graph sx-mathoverflow-c2a)

A detailed analysis of each network is provided in tables 1 and 2. As with most datasets, each network had to be preprocessed and cleaned before it could be fed into our benchmarking script. The networks used did not all have the same format, which meant that had to be reformated to the Graph Exchange XML Format (.GEXF) before Gephi could use them. Most networks explained above used adjacency lists to represent the connection between nodes. We were able to extract the networks from those adjacency lists and save them directly to the GEXF format by using the `networkx` python package. This also enabled us to perform simple analysis to the data before processing it, such as extracting the number of nodes and edges, and computing the average degree of those networks.

6 EXPERIMENTS AND RESULTS

In the baseline research, Jacomy et al. [11] state that they randomized each network three times before benchmarking them but do not explain how they were randomized. We decided to also randomize each network, using Gephi's built-in function to apply a random layout to a network. Using this operation, we only need to provide a grid size in which all nodes of a network will randomly

Dataset	Nodes	Edges	Avg. Degree
facebook_ego_0	333	5,038	30.258
facebook_ego_107	1,034	53,498	103.478
facebook_ego_348	224	6,384	57.000
facebook_ego_414	150	3,386	45.147
facebook_ego_686	168	3,312	39.429
facebook_ego_698	61	540	17.705
facebook_ego_1684	786	28,048	71.369
facebook_ego_1912	747	60,050	160.776
facebook_ego_3437	534	9,626	36.052
facebook_ego_3980	52	292	11.231
twitter_small_ego_14711172	6	8	2.667
twitter_small_ego_15053535	18	26	2.889
twitter_small_ego_15924858	10	39	7.800
twitter_small_ego_22252971	23	34	2.957
twitter_small_ego_40777046	25	39	3.120
twitter_small_ego_43858661	11	37	6.727
twitter_small_ego_96545499	13	38	5.846
twitter_small_ego_98801140	5	5	2.000
twitter_small_ego_215328630	10	33	6.600
twitter_small_ego_396721965	9	21	4.667
twitter_med_ego_9254272	155	1,779	22.955
twitter_med_ego_9460682	88	2,003	45.523
twitter_med_ego_18481292	77	1,732	44.987
twitter_med_ego_19933035	62	1,632	52.645
twitter_med_ego_21420959	91	1,787	39.275
twitter_med_ego_22106463	156	1,815	23.269
twitter_med_ego_23503181	101	1,824	36.119
twitter_med_ego_26346966	78	1,928	49.436
twitter_med_ego_163374693	164	1,749	21.329
twitter_med_ego_430313102	51	1,646	64.549
twitter_big_ego_16987303	193	13,538	140.290
twitter_big_ego_24117694	246	9,630	78.293
twitter_big_ego_89826562	216	9,715	89.954
twitter_big_ego_175553601	201	8,888	88.438
twitter_big_ego_200214366	183	9,451	103.290
twitter_big_ego_217796457	184	12,105	131.576
twitter_big_ego_248883350	184	9,042	98.283
twitter_big_ego_256497288	213	17,930	168.357
twitter_big_ego_307458983	228	9,938	87.175
twitter_big_ego_314316607	235	15,957	135.804
oregon2_010331	10,900	31,180	5.721
oregon2_010407	10,981	30,855	5.620
oregon2_010414	11,019	31,761	5.765
oregon2_010421	11,080	31,538	5.693
oregon2_010428	11,113	31,434	5.657
oregon2_010505	11,157	30,943	5.547
oregon2_010512	11,260	31,303	5.560
oregon2_010519	11,375	32,287	5.677
oregon2_010526	11,461	32,730	5.712
arxiv_condensed_matter	23,133	93,497	8.083
arxiv_general_relativity	5,242	14,496	5.531
arxiv_high_energy_physics	12,008	118,521	19.740
zacharys_karate_club	34	78	
marvel_super_heroes	10,469	178,115	34.027
c_elegans	306	2,345	7.663
yeast	2,361	7,182	6.084

Table 1: Benchmark datasets used by Jacomy et al. [11]

Dataset	Nodes	Edges	Avg. Degree
cit-HepPh	34,546	421,578	24.407
cit-HepTh	27,770	352,807	25.409
email-Enron	36,692	367,662	20.040
musae_facebook	22,470	171,002	15.220
p2p-Gnutella24	26,518	65,369	4.930
p2p-Gnutella25	22,687	54,705	4.823
sx-mathoverflow-a2q	21,688	90,489	8.345
sx-mathoverflow-c2a	13,840	81,121	11.723
sx-mathoverflow-c2q	16,836	101,329	12.037
sx-mathoverflow	24,818	239,978	19.339

Table 2: Additional Datasets Benchmarked

be placed. This led to a total of different 186 networks to benchmark.

Luckily for us, the Gephi framework already incorporated all the algorithms that we planned to benchmark. We were able to use Gephi Toolkit [9], a framework first released in 2010 (and since updated along with Gephi releases) that incorporates the Gephi modules in a standard Java library, bypassing the GUI and allowing for the execution and automation of tasks through the command-line.

Following the same principle as Jacomy et al. [11], we employed Noack's normalized^{endv} atedge inverted measure (7).

$$Q_{Noack}(p) = \frac{\frac{\sum_{\{n_1;n_2\}\in E} distance(p(n_1),p(n_2))}{|E|}}{\frac{\sum_{\{n_1;n_2\}\in N^2} distance(p(n_1),p(n_2))}{|N^2|}} \tag{6}$$

$$Q(p) = \frac{1}{Q_{Noack}(p)} \tag{7}$$

Similar to the baseline paper, we compute the layout quality at "power of 2 plus 1" step until 2049. We ran each network on a version of ForceAtlas2, ForceAtlas2 Linlog, OpenOrd, Yifan Hu and Yifan Hu Proportional. This resulted in a total of 930 results.

The algorithms were benchmarked with the following settings, inspired by settings employed by Jacomy et al. [11]:

- **ForceAtlas2** (FA2): BarnesHutTheta `1.2`; EdgeWeightInfluence `1.0`; Gravity `0.0`; JitterTolerance `1.0`; ScalingRatio `2.0`; AdjustSizes false; BarnesHutOptimize `true`; LinLogMode `false`; OutboundAttractionDistribution `false`; StrongGravityMode `false`;
- **ForceAtlas2 LinLog** (FA2_LL): BarnesHutTheta `1.2`; EdgeWeightInfluence `1.0`; Gravity `0.0`; JitterTolerance `1.0`; ScalingRatio `2.0`; AdjustSizes `false`; BarnesHutOptimize `true`; LinLogMode `true`; OutboundAttractionDistribution `false`; StrongGravityMode `false`;
- **Yifan Hu** (YH): BarnesHutTheta `1.2`; ConvergenceThreshold `1.0e-4`; InitialStep `20.797`; OptimalDistance `103.985`; StepRatio `0.95`; QuadTreeMaxLevel `10`; RelativeStrength `0.2`; AdaptiveCooling `true`;
- **Yifan Hu Proportional** (YH_P): BarnesHutTheta `1.2`; ConvergenceThreshold `1.0e-4`; InitialStep `20.797`; OptimalDistance `103.985`; StepRatio `0.95`; QuadTreeMaxLevel `10`; RelativeStrength `0.2`; AdaptiveCooling `true`;
- **OpenOrd** (OO): LiquidStage `25`; ExpansionStage `25`; CooldownStage `25`; CrunchStage `10`; SimmerStage `15`; EdgeCut `0.8`;

Upon reaching the results, we again followed the same approach as Jacomy et al. We analyse each network to find the maximum quality value, and two additional points: the Quick and Dirty point and the Quasi-Optimal point. The Quick and Dirty point is reached at 50% of maximum quality, and represents a rough estimation of the specialization. The Quasi-Optimal point is reached at 90% of the maximum quality, and should approximate a satisfying layout. The plotting of said points can be viewed in Figure 4.

Looking at the overall results reported in Figure 1, we can see that OO has the best average maximum quality over all the other algorithms, by a long margin, with FA2_LL followed slightly behind by FA2. We can see that FA2_LL, FA2, YH and YH_P have similar performance, but contrarily to the results obtained by Jacomy et al, we observe that YH and YH_P both outperform FA2 on the Quasi-Optimal measure, and outperform FA2 for the Quick and Dirty measure.

We can see from Figure 2 each algorithm's performance on each network by plotting the quasi-optimal measure over the number of nodes a network possesses. We find that for a majority of larger networks, FA2_LL's performance is very close to YH's performance, if not better.

In figure 3, we show the different algorithms yield a different layout. And while we can more easily determine the clusters in the smaller network (facebook_414), this is harder to view from the other networks.

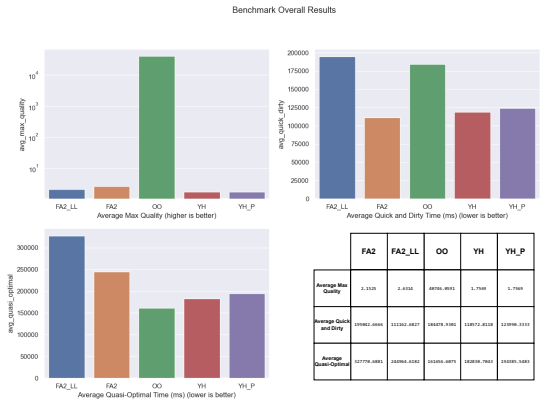

Figure 1: Benchmark Overall Results

The benchmark was run on an AMD Ryzen 5 3600 CPU and 16Gb of 3200Mhz RAM. To enable the best reproducibility, the code can be found in the project repository [37][1].

[1]https://github.com/ernestvmo/forceatlas2-SNACS

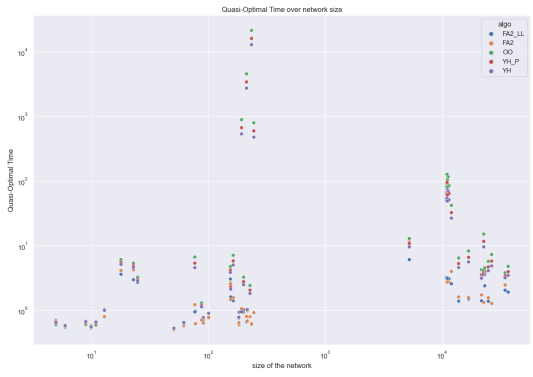

Figure 2: Quai-Optimal over Time

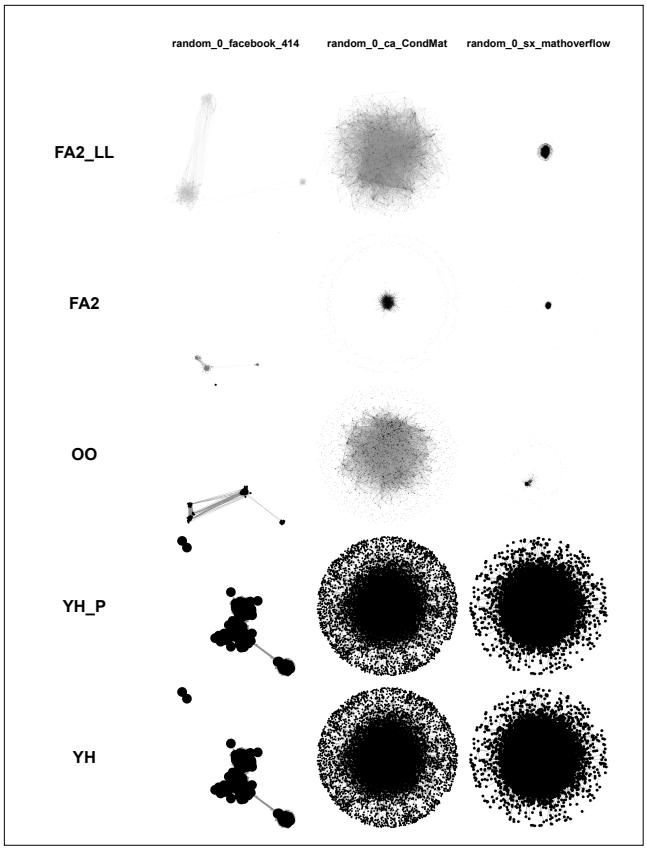

Figure 3: Example of Different Layouts

7 CONCLUSION AND FUTURE WORK

In conclusion, we find the Noack measure 7 employed by Jacomy et al to be a convenient measure to evaluate ForceAtlas2 variants and Yifan Hu variants, but a very poor measure to evaluate the performance of OpenOrd. Even tho OpenOrd ended up outperforming all algorithms, its multiple-stage mechanism lead to inconsistent measures during the benchmark. We also join in the opinion of Jacomy et al [11] that the measure fails to capture the clusters that are visible in the visualized network, and a more sophisticated measure should be investigated for future comparisons. We also find that the performance of Gephi does not scale well the larger the network gets, with averages of over an hour to benchmark a network of over 20,000 nodes on a single algorithm. This unfortunate disadvantage will lead users to consider other frameworks such as GraphViz [5] to visualize larger networks.

REFERENCES

[1] Josh Barnes and Piet Hut. 1986. A hierarchical O(N log N) force-calculation algorithm. 324, 6096 (Dec. 1986), 446–449. https://doi.org/10.1038/324446a0
[2] Mathieu Bastian, Sebastien Heymann, and Mathieu Jacomy. 2009. Gephi: an open source software for exploring and manipulating networks. In *Proceedings of the international AAAI conference on web and social media*, Vol. 3. 361–362.
[3] Dongbo Bu, Yi Zhao, Lun Cai, Hong Xue, Xiaopeng Zhu, Hongchao Lu, Jingfen Zhang, Shiwei Sun, Lunjiang Ling, Nan Zhang, Guo-Jie Li, and Runsheng Chen. 2003. Topological structure analysis of the protein-protein interaction network in budding yeast. *Nucleic acids research* 31 (06 2003), 2443–50. https://doi.org/10.1093/nar/gkg340
[4] Peter Eades. 1984. A heuristic for graph drawing. *Congressus numerantium* 42 (1984), 149–160.
[5] John Ellson, Emden Gansner, Lefteris Koutsofios, Stephen C. North, and Gordon Woodhull. 2002. Graphviz— Open Source Graph Drawing Tools. In *Graph Drawing*, Petra Mutzel, Michael Jünger, and Sebastian Leipert (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 483–484.
[6] Thomas MJ Fruchterman and Edward M Reingold. 1991. Graph drawing by force-directed placement. *Software: Practice and experience* 21, 11 (1991), 1129–1164.
[7] Thomas M. J. Fruchterman and Edward M. Reingold. 1991. Graph drawing by force-directed placement. *Software: Practice and Experience* 21, 11 (1991), 1129–1164. https://doi.org/10.1002/spe.4380211102 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/spe.4380211102
[8] Johannes Gehrke, Paul Ginsparg, and Jon Kleinberg. 2003. Overview of the 2003 KDD Cup. *SIGKDD Explor. Newsl.* 5, 2 (dec 2003), 149–151. https://doi.org/10.1145/980972.980992
[9] Gephi. [n. d.]. Gephi Toolkit. https://gephi.org/toolkit/.
[10] Yifan Hu. 2005. Efficient and High Quality Force-Directed Graph Drawing. *The Mathematica Journal* 10 (01 2005), 37–71.
[11] Mathieu Jacomy, Tommaso Venturini, Sebastien Heymann, and Mathieu Bastian. 2014. ForceAtlas2, a Continuous Graph Layout Algorithm for Handy Network Visualization Designed for the Gephi Software. *PLOS ONE* 9, 6 (06 2014), 1–12. https://doi.org/10.1371/journal.pone.0098679
[12] Jure Leskovec. [n. d.]. Autonomous systems - Oregon-2. http://snap.stanford.edu/data/Oregon-2.html.
[13] Jure Leskovec. [n. d.]. Condense Matter collaboration network. http://snap.stanford.edu/data/ca-CondMat.html.
[14] Jure Leskovec. [n. d.]. Enron email network. http://snap.stanford.edu/data/email-Enron.html.
[15] Jure Leskovec. [n. d.]. Facebook Large Page-Page Network. http://snap.stanford.edu/data/facebook-large-page-page-network.html.
[16] Jure Leskovec. [n. d.]. General Relativity and Quantum Cosmology collaboration network. http://snap.stanford.edu/data/ca-GrQc.html.
[17] Jure Leskovec. [n. d.]. Gnutella peer-to-peer network, August 24 2002. http://snap.stanford.edu/data/p2p-Gnutella24.html.
[18] Jure Leskovec. [n. d.]. Gnutella peer-to-peer network, August 25 2002. http://snap.stanford.edu/data/p2p-Gnutella25.html.
[19] Jure Leskovec. [n. d.]. High Energy Physics - Phenomenology collaboration network. http://snap.stanford.edu/data/ca-HepPh.html.
[20] Jure Leskovec. [n. d.]. High-energy physics citation network. http://snap.stanford.edu/data/cit-HepPh.html.
[21] Jure Leskovec. [n. d.]. High-energy physics theory citation network. http://snap.stanford.edu/data/cit-HepTh.html.
[22] Jure Leskovec. [n. d.]. Math Overflow temporal network. http://snap.stanford.edu/data/sx-mathoverflow.html.
[23] Jure Leskovec. [n. d.]. Social circles: Facebook. http://snap.stanford.edu/data/ego-Facebook.html.
[24] Jure Leskovec. [n. d.]. Social circles: Twitter. http://snap.stanford.edu/data/ego-Twitter.html.
[25] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. 2005. Graphs over Time: Densification Laws, Shrinking Diameters and Possible Explanations. In *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining* (Chicago, Illinois, USA) *(KDD '05)*. Association for Computing Machinery, New York, NY, USA, 177–187. https://doi.org/10.1145/1081870.1081893

[26] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. 2007. Graph Evolution: Densification and Shrinking Diameters. *ACM Trans. Knowl. Discov. Data* 1, 1 (mar 2007), 2–es. https://doi.org/10.1145/1217299.1217301

[27] Jure Leskovec and Andrej Krevl. 2014. SNAP Datasets: Stanford Large Network Dataset Collection. http://snap.stanford.edu/data.

[28] Jure Leskovec, Kevin J. Lang, Anirban Dasgupta, and Michael W. Mahoney. 2008. Community Structure in Large Networks: Natural Cluster Sizes and the Absence of Large Well-Defined Clusters. https://doi.org/10.48550/ARXIV.0810.1355

[29] Marcus Bingenheimer. [n. d.]. Gephi Datasets. https://github.com/gephi/gephi/wiki/Datasets.

[30] Shawn Martin, W. Brown, Richard Klavans, and Kevin Boyack. 2011. OpenOrd: An Open-Source Toolbox for Large Graph Layout. *Proc SPIE* 7868 (01 2011), 786806. https://doi.org/10.1117/12.871402

[31] Julian McAuley and Jure Leskovec. 2012. Learning to Discover Social Circles in Ego Networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1* (Lake Tahoe, Nevada) *(NIPS'12)*. Curran Associates Inc., Red Hook, NY, USA, 539–547.

[32] Andreas Noack. 2007. Energy models for graph clustering. *J. Graph Algorithms Appl.* 11, 2 (2007), 453–480.

[33] Andreas Noack. 2007. *Unified quality measures for clusterings, layouts, and orderings of graphs, and their application as software design criteria.* Ph. D. Dissertation. BTU Cottbus-Senftenberg.

[34] Andreas Noack. 2009. Modularity clustering is force-directed layout. *Physical Review E* 79, 2 (2009), 026102.

[35] Ashwin Paranjape, Austin R. Benson, and Jure Leskovec. 2017. Motifs in Temporal Networks. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*. ACM. https://doi.org/10.1145/3018661.3018731

[36] Benedek Rozemberczki, Carl Allen, and Rik Sarkar. 2019. Multi-scale Attributed Node Embedding. https://doi.org/10.48550/ARXIV.1909.13021

[37] Ernest Vanmosuinck and Mouni Priyanka. 2022. *Benchmarking the performance of ForceAtlas2 and other algorithms on Large Networks.*

[38] Duncan J. Watts and Steven H. Strogatz. 1998. Collective dynamics of 'small-world'networks. *Nature* 393, 6684 (1998), 440–442. https://doi.org/10.1038/30918

[39] Wayne W Zachary. 1977. An information flow model for conflict and fission in small groups. *Journal of anthropological research* 33, 4 (1977), 452–473.

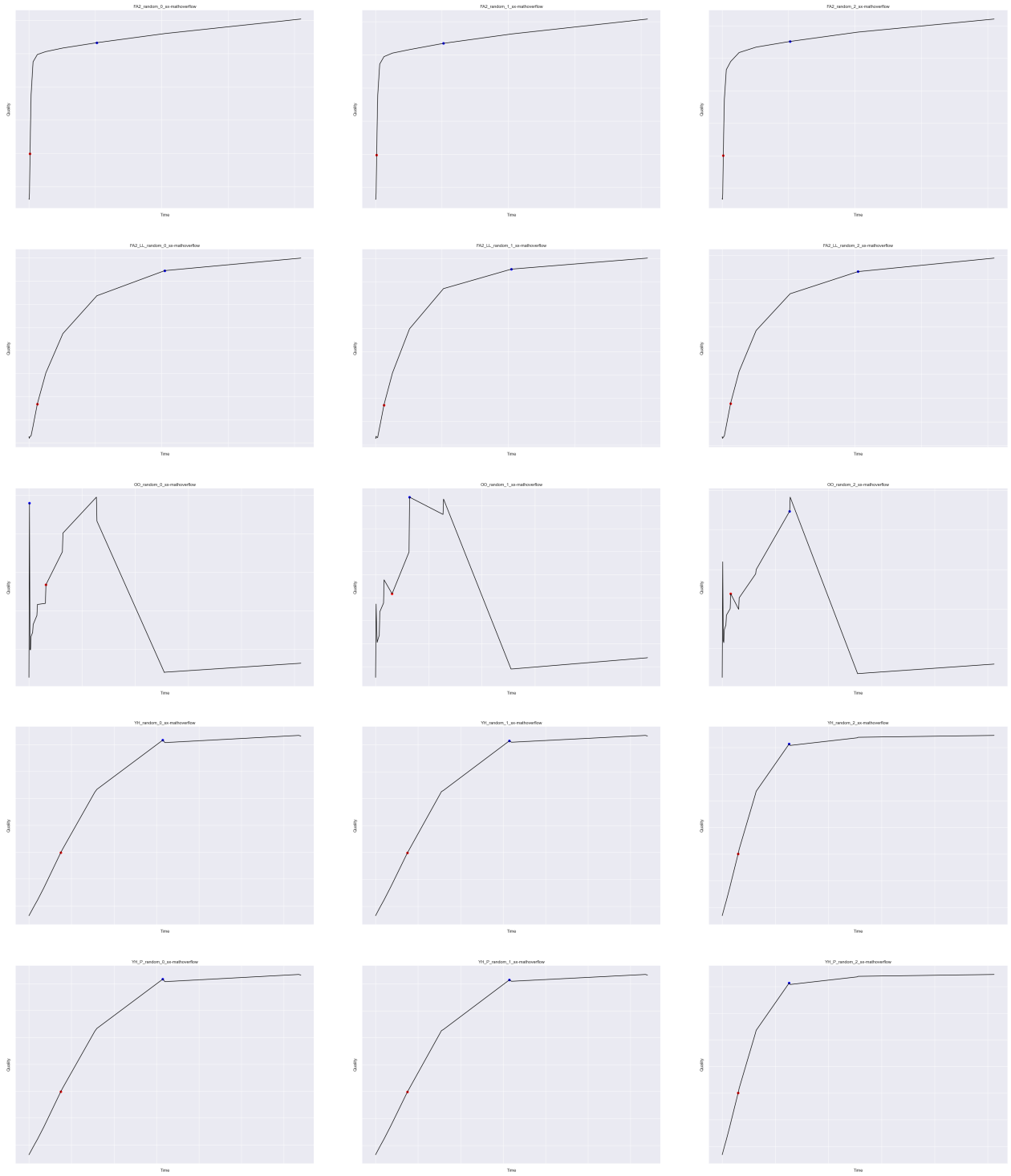

Figure 4: Quick and Dirty and Quasi-Optimal points