

Force-directed Voronoi-based Method for Graph Drawing in Constrained Boundaries

Fernando Wong
Allm Inc.
Email: f.wong@allm.net

Horacio Sanson
Allm Inc.
Email: horacio@allm.net

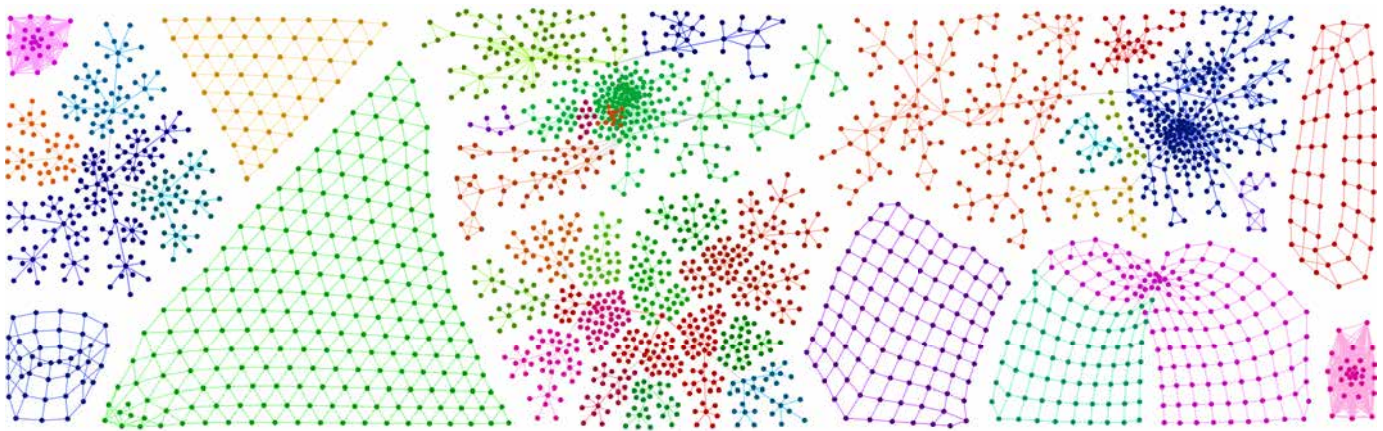


Fig. 1: A rectangular layout of a multicomponent graph generated with our approach.

Abstract—We present an efficient force-directed method for evolving an initial graph layout in order to better visualize it, while at the same time distributing graph vertices so as to make full use of the available drawing space. This results in layouts that adjust themselves to the specified boundaries of the drawing area. Previous works have solely focused on the placement of graph vertices and edges for visualization purposes, working under the assumptions of no area limitations nor boundary constraints. Similar to the process of computing a centroidal Voronoi diagram from a set of points, our algorithm iteratively computes a Voronoi diagram over the set of graph vertices, and attracts them towards the centroids of their corresponding Voronoi cells at each iteration. At the same time, attractive and repulsive forces are applied between adjacent vertices in the graph, in order to place them closer to each other in the final layout. Additional consideration is given in order to prevent overlapping in graphs with multiple components.

I. INTRODUCTION

Graphs are important data structures that represent the relationships between multiple entities, such as people, computers, documents, among others. Particularly when there is a high volume of interconnected data, there might be many hidden relationships that can only be clearly understood through its corresponding graph representation, and thus, developing techniques that help us to better visualize these connections is an important topic of research.

In this work, we present a simple and efficient force-directed graph layout algorithm based on centroidal Voronoi diagrams, which has the particular property of expanding the graph across an arbitrary boundary, making effective use of the

available drawing space. This method is capable of generating graph layouts for boundaries of any shape, provided that the specified boundary for the layout is convex and a clipped Voronoi diagram can be computed over it.

Our approach has the following advantages over conventional force-directed graph layout algorithms:

- **Reduced time complexity:** Conventional algorithms often rely on the computation of n -body simulations, which involves $O(n^2)$ time complexity, with n being the number of vertices, unless approximations are employed [1]. Instead, our approach calculates forces across graph edges and between vertices and centroids in a Voronoi diagram, which can be computed in $O(n \log n)$ time [2].
- **Effective use of space:** Due to the nature of centroidal Voronoi diagrams, our graphs tend to fully cover the available space. This could be particularly useful for printing purposes, for example, as a large graph might not fit into a single page unless it is significantly scaled.
- **Usable as a building block for complex visualizations:** Since the available space for displaying a graph can be constrained in our approach, we can use this property to build more elaborate visualizations, such as bar charts and pie charts with embedded graph data.
- **Ease of implementation:** The algorithm does not involve complex operations, except for the computation of Voronoi diagrams, for which fast and efficient algorithms exist and are readily available in several computational geometry libraries in a variety of programming languages.

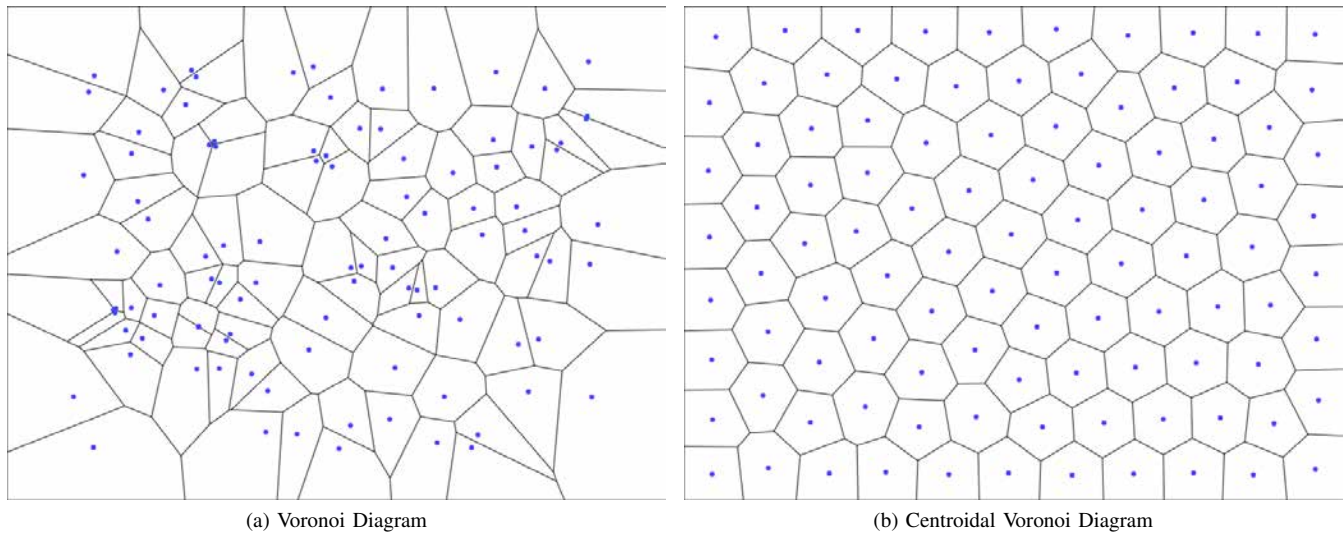


Fig. 2: Voronoi diagrams. Each cell denotes the area of the space that is closest to its generator point than to any other.

To the best of our knowledge, this paper presents the first graph drawing algorithm that attempts to spread a graph throughout a given delimited boundary.

II. BACKGROUND AND RELATED WORK

A. Force-directed Graph Layout Algorithms

Among all graph drawing techniques, force-directed methods are the ones that tend to produce the most aesthetically pleasing graph layouts, while also allowing the animation of the entire layout process.

These methods often rely in physical simulations of spring systems [3], [4] and particle interactions [5], providing a simple means of generating good-quality graph drawings, at the cost of increased computation times. This is mostly due to these approaches requiring expensive n -body simulations, which involve calculating repulsive forces between all pairs of vertices in the graph. However, approximation techniques such as the Barnes-Hut simulation can be used in order to accelerate the process [1], reducing its time complexity from $O(n^2)$ to $O(n \log n)$.

Despite their popularity, conventional force-directed methods have been difficult to apply to graphs of considerable size, mostly due to the above-mentioned computational cost. For this reason, there has been an increased interest in researching multiscale [6]–[9], distributed [10], parallel [11], and other methods that make possible the application of force-directed algorithms to graphs with thousands of vertices.

We recommend reading Kobourov's survey on force-directed graph layout algorithms for a deeper introduction into this topic [12].

B. Centroidal Voronoi Diagrams

Given a set of generator points $P = \{p_0, \dots, p_n\}$, a Voronoi diagram is a subdivision of the space into a group of closely fitted polygons or cells $C = \{c_0, \dots, c_n\}$, for which each cell

c_i contains all points closest to point p_i (Figure 2a). It has been demonstrated that Voronoi diagrams can be efficiently constructed in $O(n \log n)$ time [2], and can be computed in two or more dimensions, as well as in Euclidean, hyperbolic, and other spaces.

The Voronoi diagram and the Delaunay triangulation, its dual problem, have seen numerous applications on the fields of geographical information systems and computer graphics, for example, which often have a need for nearest-neighbor queries.

A centroidal Voronoi diagram is a special case of the Voronoi diagram for which each generator point p_i also happens to be the centroid of its corresponding cell c_i . This type of diagrams often resemble a hexagonal grid as shown in Figure 2b. It is known that a regular Voronoi diagram can be transformed into a centroidal Voronoi diagram through Lloyd's relaxation [13]. The process consists in iteratively computing the Voronoi diagram of the set of points, shifting each point to the centroid of its corresponding cell, and repeating the process on the set of shifted points. This is repeated until reaching convergence or until a specified number of iterations have been completed.

There have been studies involving Voronoi diagrams and the drawing of tree structures [14], and in recent years, centroidal Voronoi diagrams have found useful applications for tree visualization in the form of Voronoi treemaps [15], [16]. On the other hand, to our knowledge, there is no prior work on the application of Voronoi diagrams for the visualization of general graph structures.

III. PROPOSED METHOD

A. Overview

Our approach is loosely based on Lloyd's algorithm for generating centroidal Voronoi diagrams [13], and prior work on force-directed graph layouts. Contrary to previous methods,

our approach applies attraction-repulsion forces along edges of the graph, and gravitational forces to each vertex towards the centroid of its Voronoi cell. No n-body simulations are involved during the computation.

In summary, for a given graph $G = \{V, E\}$ over a set of vertices V and edges E , our algorithm is as follows:

- 1) Place each vertex $v \in V$ in the drawing area.
- 2) For each edge $(u, v) \in E$, compute attraction and repulsion forces $F_a(u, v)$ and $F_r(u, v)$ between vertices u and v (Equations 1 and 2).
- 3) Compute the Voronoi diagram of the set of vertices V .
- 4) For each $v \in V$, compute a gravitational force $F_g(v)$ that attracts v towards the centroid of its corresponding Voronoi cell C_v (Equation 4).
- 5) Move each vertex v in the direction defined by the sum of all computed forces $F_a(u, v)$ and $F_r(u, v)$ for each neighbor u of v , and $F_g(v)$ towards the centroid of its Voronoi cell.
- 6) Repeat from Step 2, until reaching convergence or a specified number of iterations have been completed.

In general, following these steps works well for graphs with a single component, i.e. graphs for which there is a path between any pair of vertices u and v . For the multiple component case, we often apply repulsive forces between components as well, in order to avoid overlapping to a certain degree (Section III-D).

B. Attractive-Repulsive Forces

As previously mentioned, we apply attraction and repulsion forces between vertices sharing an edge. For a given edge $(u, v) \in E$, the force attracting vertex u to v is defined as:

$$F_a(u, v) = \frac{w_a}{u_d + v_d} \cdot \frac{\|\mathbf{v} - \mathbf{u}\|^2}{k} \cdot \overrightarrow{\mathbf{v} - \mathbf{u}} \quad (1)$$

and the force repelling u from v as:

$$F_r(u, v) = \frac{w_r}{u_d + v_d} \cdot \frac{k^2}{\|\mathbf{u} - \mathbf{v}\|} \cdot \overrightarrow{\mathbf{u} - \mathbf{v}} \quad (2)$$

where $\mathbf{v}$ is the vector of coordinates for vertex v in Euclidean space, $\|\mathbf{v}\|$ denotes the magnitude of $\mathbf{v}$, $\overrightarrow{\mathbf{v}}$ is the normalized vector of $\mathbf{v}$, v_d is the degree of vertex v , w_a and w_r are weighting factors for their respective forces, and k is the optimal distance between vertices in the layout.

As vertices with a high degree (i.e. with several neighbors) attract an increased number of vertices, resulting in multiple vertices being placed too close to each other, we inversely scale these forces by $u_d + v_d$ in order to weaken their influence over vertices with a high number of connections.

The distance k is defined as in previous works [5]:

$$k = \sqrt{\frac{A}{|V|}} \quad (3)$$

for which A denotes the total available drawing area and $|V|$ is the total number of vertices in V .

C. Gravitational Forces

At each iteration of the algorithm, we compute a Voronoi diagram based on the current coordinates of graph vertices in the layout, and then each vertex is pulled by gravitational forces towards the centroid of their corresponding Voronoi cell in the diagram. This is performed in order to distribute the graph nodes across the available space.

Ideally, this gravitational force should be proportional to the distance between each vertex and its corresponding cell centroid. Also, as vertices with more edges would be subject to more forces being applied to them, the gravitational force should also be proportional to the degree of a given vertex in order to compensate for the additional force interaction.

Taking this into account, we define the gravitational force applied to each vertex v as:

$$F_g(v) = w_g \cdot \max(v_d, 1) \cdot \frac{\|\mathbf{C}_v - \mathbf{v}\|^2}{k} \cdot \overrightarrow{\mathbf{C}_v - \mathbf{v}} \quad (4)$$

where $\mathbf{C}_v$ denotes the centroid of Voronoi cell C_v .

D. Multiple Component Graphs and Inter-component Forces

Applying the above-mentioned forces will produce good results in general, although for some cases the layout algorithm could overlap multiple graph components if the graph is disconnected, that is, there are two or more subgraphs $G_1, \dots, G_n$ in graph G for which there is no path between any pair of vertices $v_i \in G_i$ and $v_j \in G_j$ given that $G_i \neq G_j$. This is mainly due to attractive forces not being strong enough to overcome the gravitational forces pulling each vertex towards its corresponding centroid in the diagram. The problem is more prominent in graphs containing many isolated vertices ($v_d = 0$), as in the one shown in Figure 3, as these are pulled by gravitational forces only.

To alleviate this problem, we apply an additional force to each vertex, which repels them from neighboring vertices not belonging to the same connected component. For this purpose, we first determine to which component each vertex belongs to, in a preprocessing step that is executed only once before the iterative application of forces begins.

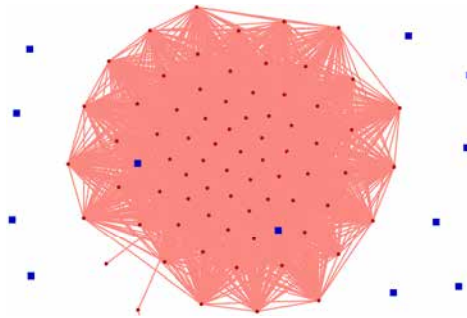


Fig. 3: Two isolated vertices (squares) end up stuck inside a graph component on a layout generated with $w_c = 0$. The vertices cannot get out of the component, as their gravitational force is too strong and no other forces act on them.

Then, at each iteration of the algorithm, we compute the center of each connected component of the graph, and apply a repulsive force to each vertex so that it gets repelled from graph components it does not belong to. Let $G_v = \{V_v, E_v\}$ denote the graph component to which vertex v belongs to, for which $G_v \subseteq G$, then the center $\mathbf{G}_v$ of component G_v is simply computed as:

$$\mathbf{G}_v = \frac{1}{|V_v|} \sum_{u \in V_v} \mathbf{u} \quad (5)$$

The inter-component repulsive force applied to vertex v is then defined as:

$$F_c(v) = w_c \sum_{u \in N_v} |V_u| \cdot h(u, v) \cdot f_c(u, v) \quad (6)$$

$$f_c(u, v) = \frac{\max(0, r(G_u) - \|\mathbf{v} - \mathbf{G}_u\|)^2}{\|\mathbf{v} - \mathbf{G}_u\|} \cdot \overrightarrow{\mathbf{G}_v - \mathbf{G}_u} \quad (7)$$

for which N_v is the set of neighbors of vertex v in the Voronoi diagram, $h(u, v)$ is a binary function for which:

$$h(u, v) = \begin{cases} 1, & \text{if } G_u \neq G_v \\ 0, & \text{otherwise} \end{cases} \quad (8)$$

and $r(G_u)$ is an estimate of the minimum distance that should separate any vertices $v \notin V_u$ from $\mathbf{G}_u$, which is computed as:

$$r(G_u) = \sqrt{\frac{\sum_{u \in V_u} \min(A(C_u), k^2)}{\pi}} \quad (9)$$

for which $A(C_u)$ is the area of the Voronoi cell C_u .

Please note that $\overrightarrow{F_c(v)}$ is applied in the direction of vector $\overrightarrow{\mathbf{G}_v - \mathbf{G}_u}$ and not $\mathbf{v} - \mathbf{G}_u$, that is, the force is applied away from the center of component G_u towards the center of G_v .

By applying these forces, vertices are repelled from nearby graph components, thus avoiding component overlapping to a certain extent, and improving the overall visualization of the different components of the graph.

IV. RESULTS AND DISCUSSION

As the prototype of our graph layout method was meant to be used in a web application, our graph layout algorithm was implemented in Javascript as a plugin for Linkurious.js [17], a popular fork of the Sigma.js graph drawing library [18].

Voronoi diagrams were computed with a publicly available Javascript implementation of Fortune's algorithm [19], which we modified so that it could handle the computation of circular Voronoi diagrams as well for the purposes of testing.

At the time of writing this paper we could not find previous research on graph layout algorithms with boundary constraints, thus, in order to evaluate our approach, we ran multiple graph cases against both our algorithm and the Fruchterman-Reingold (FR) algorithm [5] implementation provided in Linkurious.js. This implementation modifies the original FR algorithm so that an additional gravitational force can be specified, which attracts nodes towards the center of the drawing space, producing a graph layout that is mostly circular if the gravity is strong enough, and preventing disconnected components from being overly separated due to the action of repulsive forces. We modified the FR implementation provided in the library so that it runs faster, as the original would not run for graphs having over 500 nodes in a reasonable amount of time.

These tests were conducted on a MacBook Pro with a 2.9 Ghz Intel Core i5 processor and 8 GB of RAM on the Google Chrome web browser. All results shown in this section were generated with parameters $w_a = 10$, $w_r = 1$, $w_g = 20$ and $w_c = 5$ for our method, and with $gravity = 1$ for the FR algorithm results. Table I shows a summary of the test results in terms of processing time, iterations and number of crossings in the final graph layout.

In most cases, the FR algorithm achieves a relatively uniform distribution of vertices across the drawing space. While this might be a desirable aesthetic trait, in constrained drawing spaces this could actually hinder the visualization. In contrast, vertices are not evenly separated in our approach, making it easier to identify clusters in general, although vertex clustering is noticeable in some cases. Having less distance between adjacent vertices enhances the visualization of graph clusters in general, and often reduces the number of edge crossings, which are known to degrade the final graph layout [20]. In addition, our algorithm converges in a much reduced amount of time and iterations in general, as shown in Table I.

Figures 4a and 4d show examples of a tree graph laid out with both approaches. While the FR algorithm exhibits a more even distribution of graph vertices across the drawing space, this results in an increased amount of crossings and makes it slightly harder to discern branches of the tree.

Both approaches present issues when laying out grid-like

TABLE I: Comparison between Fruchterman-Reingold graph layouts and our approach.

			Fruchterman-Reingold (FR)				Ours			
Graph	Vertices	Edges	Time (T)	Iterations (I)	T/I	Crossings	Time (T)	Iterations (I)	T/I	Crossings
Tree	163	162	47.40s	13,206	3.59ms	38	2.53s	4,402	0.57ms	14
Quad Grid	400	760	79.26s	20,119	3.94ms	540	3.48s	3,162	1.10ms	468
Triangular Grid	465	1,305	118.75s	29,016	4.09ms	643	1.97s	1,550	1.27ms	2,243
Random Graph	1,000	1,467	113.54s	19,003	5.97ms	1,214	130.09s	54,219	2.40ms	758
Multicomponent	2,495	4,324	1093.47s	61,938	17.65ms	29,163	368.09s	56,854	6.47ms	25,554
Facebook Circles	4,039	88,234	165.62s	3,751	44.15ms	61,617,901	228.90s	12,648	18.10ms	56,060,491

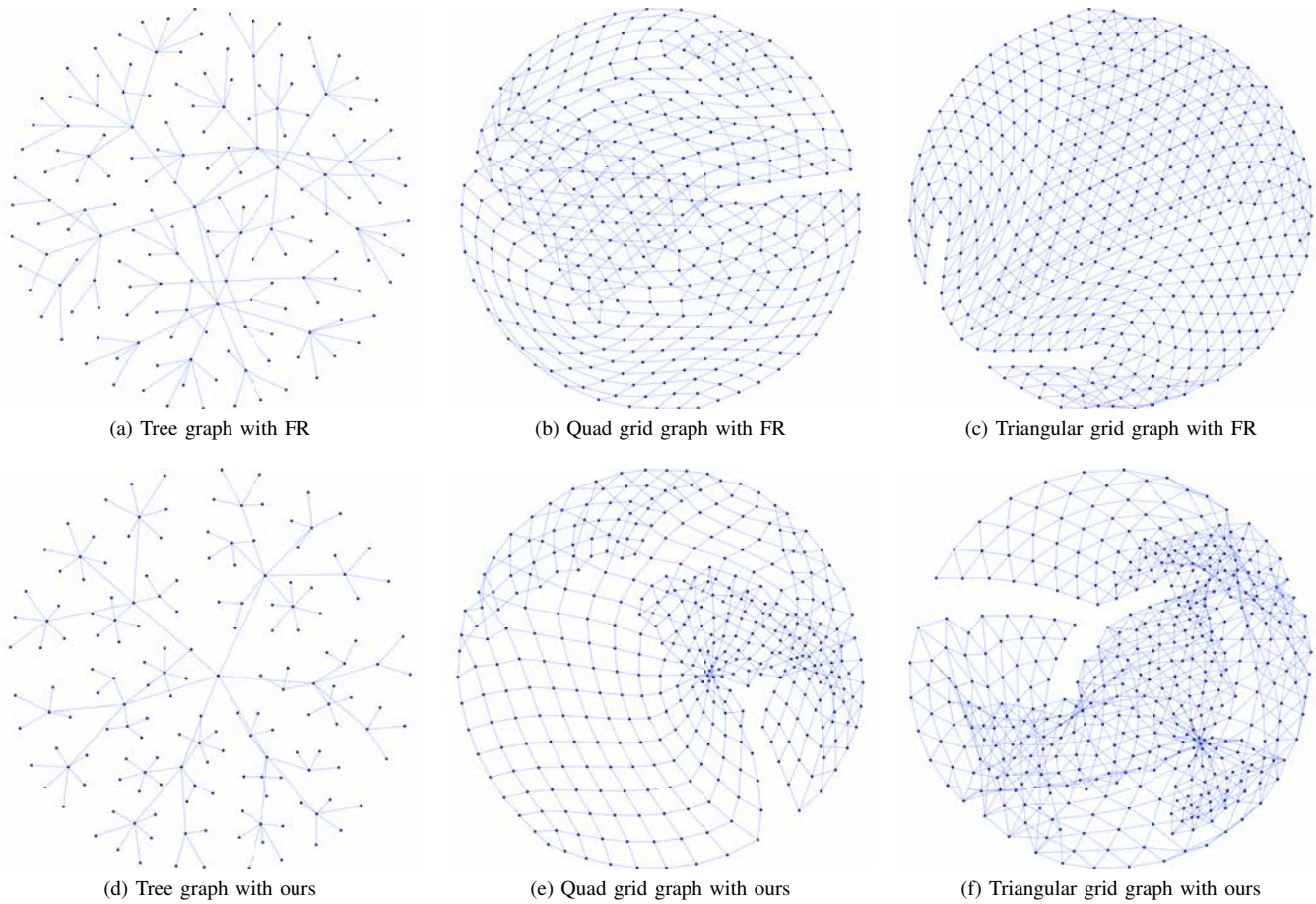


Fig. 4: Circular layouts of simple graphs produced by the Fruchterman-Reingold (FR) algorithm and our approach.

graphs as can be seen in Figures 4b and 4e, which show layouts for a uniform quad grid graph, and in Figures 4c and 4f, which show a triangular one. This is due to the additional gravity and boundary constraints preventing the graph from properly expanding. We find that, in general, FR does a better job at expanding the graph in these cases, although it would be more appropriate to use the original FR algorithm, without gravity constraints, for laying out these types of graphs.

Figures 5a and 5d show layouts for a graph randomly generated through a “Forest Fire” graph evolution model [21]. Both algorithms show similar results in this case, with FR showing relatively uniform separation between vertices as in previous examples. Certain details such as clusters and leaf nodes are difficult to visualize due to the even separation of vertices, while these features are more noticeable in our approach.

Figures 5b and 5e show a graph containing multiple components with different topologies. The FR algorithm better expands graph components with grid-like topologies, while overlapping disconnected components in some cases as shown in Figure 5b. On the other hand, our approach does not present overlappings due to the inter-component forces introduced in

Section III-D, however, the components themselves usually fail to be expanded completely as shown in the figure.

Figures 5c and 5f show layouts for a 4000+ vertex graph of anonymized Facebook circle data taken from the Stanford Large Network Dataset Collection [22]. For the purposes of visualization, graph clusters were computed beforehand with the Louvain community detection method [23] and drawn with different colors in the final layouts. The strength of repulsive and attractive forces in a graph of this size allows for very little movement of the vertices, resulting in the FR algorithm converging in a greatly reduced amount of iterations in this case. As can be seen, with such an increased amount of vertices and edges, the FR algorithm can no longer produce an evenly-distributed circular graph layout like in previous examples, and relatively small graph clusters are difficult to identify. In contrast, such clusters can be seen with the layout produced by our approach while still retaining the circular boundary constraint.

Layouts produced by the FR algorithm could be made more circular by increasing the *gravity* parameter as in Figure 6, although the final layout might not be entirely circular. Please note, however, that the main purpose of the FR algorithm with gravitational forces is not to produce a circular layout,

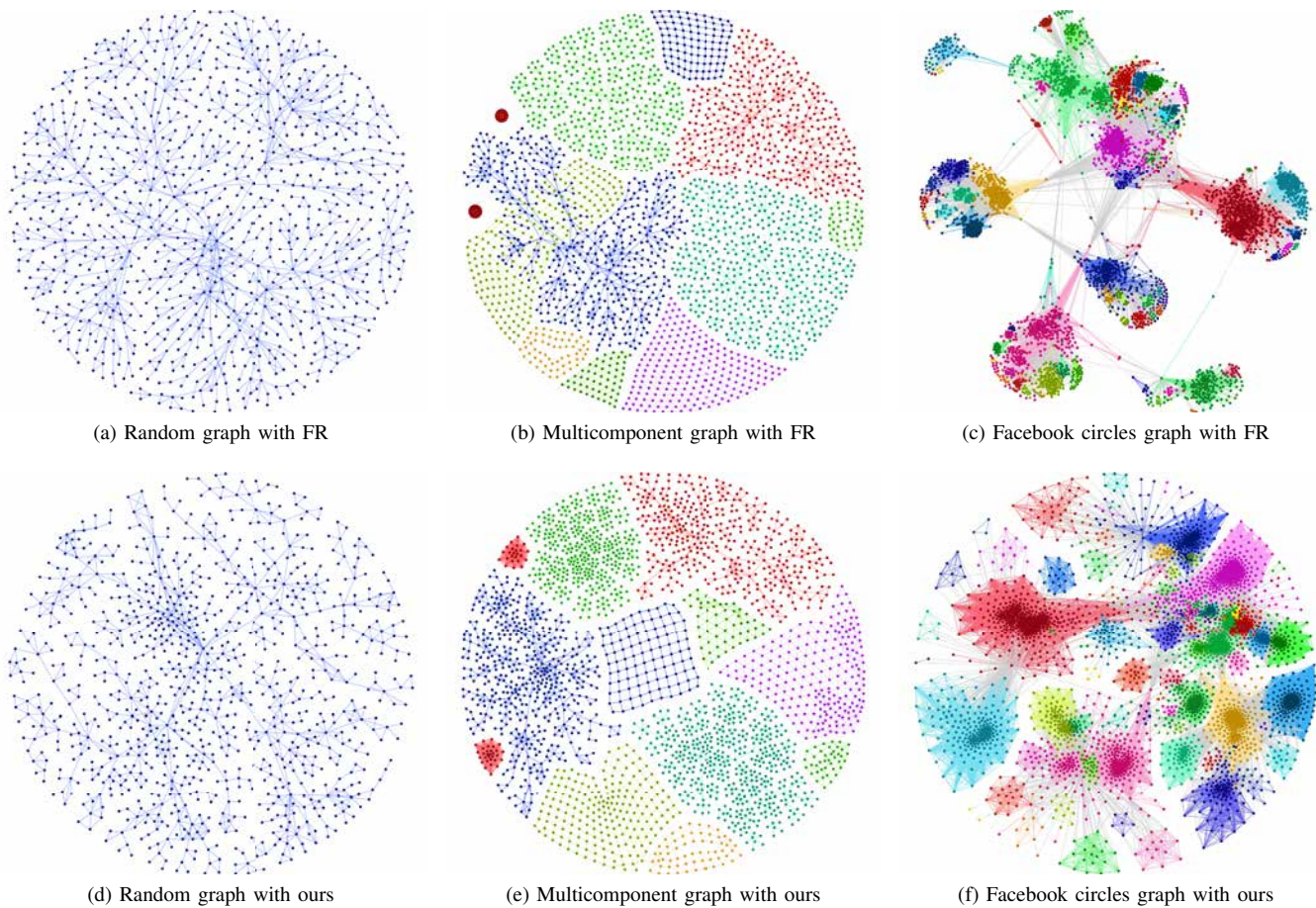


Fig. 5: Circular layouts of more complex graphs produced by the Fruchterman-Reingold (FR) algorithm and our approach.

but to prevent graph components in disconnected graphs from straying too far away from each other due to the action of repulsive forces. On the other hand, our algorithm is designed so that the graph is expanded throughout a specified delimited boundary, and is not limited to circular boundaries as demonstrated in Figures 1 and 7.

V. CONCLUSION AND FUTURE WORK

An efficient force-directed graph layout method based on centroidal Voronoi diagrams has been introduced in this paper. Our algorithm fully expands the graph throughout the available drawing space, while keeping neighboring vertices close to each other, facilitating the visualization of clusters present in the graph. Since the layout algorithm occupies the entire specified drawing area, it is suitable for building more complex data visualizations with embedded graph data. The algorithm works well for single and multiple component graphs, and an improvement has been proposed to better visualize different components in the multi-component case.

As forces are applied iteratively to graph vertices, the overall layout is improved at each iteration. While this iterative approach produces good layout results in general, as in previous force-directed approaches, the end result greatly

depends on the initial position of the graph vertices, and it often fails to find the best possible graph layout that minimizes edge crossings while keeping linked vertices close to each other. Computing such a layout is an NP-hard problem but, nonetheless, we would like to explore methods to improve the vertex positioning in order to avoid unnecessary edge crossings and speed up the process by, for example, applying ideas from multiscale methods [6]–[9].

Although our algorithm expands the graph across the drawing space, this is only feasible as long as the space is convex in shape, otherwise, the attractive-repulsive forces applied to the vertices could displace them outside of the specified boundaries. We would like to extend our approach to handle cases with arbitrary space constraints, such as holes in the drawing areas and concave boundaries.

We would also like to evaluate the use of power diagrams instead of Voronoi diagrams in order to control the separation between graph vertices in the final visualization, in the hopes of giving more focus to specific vertices, in a similar way as done in Voronoi treemaps for controlling the size of tree nodes based on their weights [15].

Since our method is similar to Lloyd's algorithm for computing centroidal Voronoi diagrams, we believe it is

possible to reduce the time it takes the algorithm to converge through the use of techniques introduced in previous works that aim to increase the performance of centroidal Voronoi diagram computation, by employing advanced minimization methods such as L-BFGS [24] and leveraging the power of dedicated graphics hardware [25].

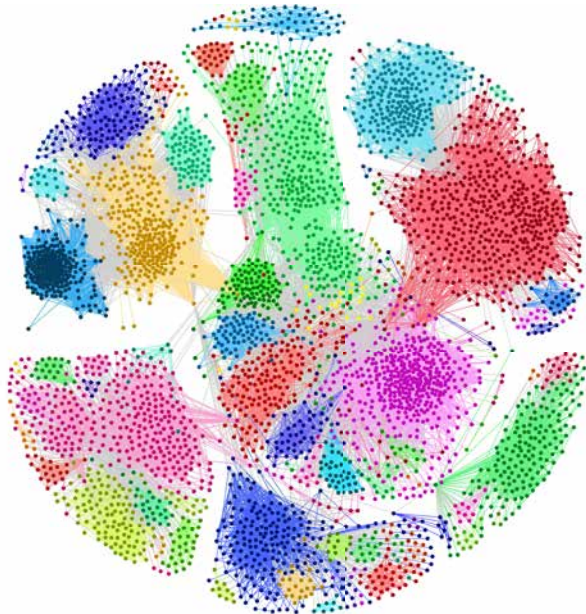


Fig. 6: Facebook circles graph with FR algorithm and $gravity = 10$.

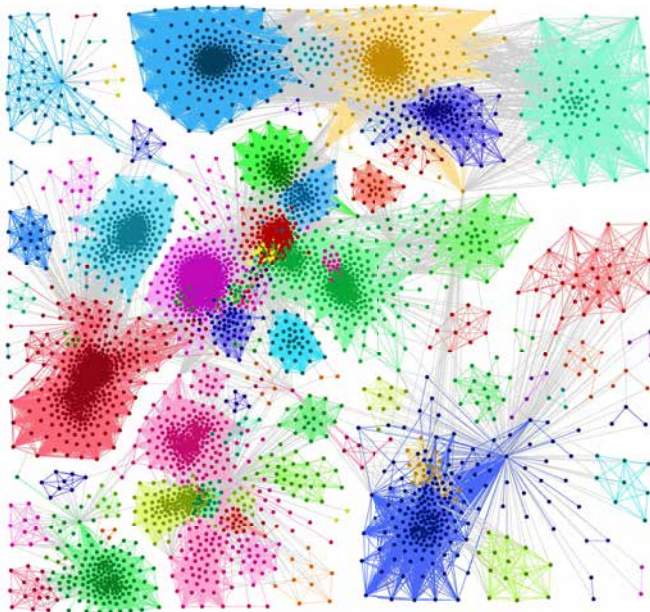


Fig. 7: Facebook circles graph with our approach in a rectangular boundary.

REFERENCES

- [1] J. Barnes and P. Hut, "A Hierarchical $O(N \log N)$ Force-Calculation Algorithm," *Nature*, vol. 324, pp. 446–449, 1986.
- [2] S. Fortune, "A Sweep-line Algorithm for Voronoi Diagrams," in *Proceedings of the Second Annual Symposium on Computational Geometry*, ser. SCG '86. New York, NY, USA: ACM, 1986, pp. 313–322.
- [3] P. Eades, "A Heuristic for Graph Drawing," *Congressus Numerantium*, vol. 42, pp. 149–160, 1984.
- [4] T. Kamada and S. Kawai, "An Algorithm for Drawing General Undirected Graphs," *Inf. Process. Lett.*, vol. 31, no. 1, pp. 7–15, Apr. 1989.
- [5] T. M. J. Fruchterman and E. M. Reingold, "Graph Drawing by Force-directed Placement," *Softw. Pract. Exper.*, vol. 21, no. 11, pp. 1129–1164, Nov. 1991.
- [6] D. Harel and Y. Koren, "A Fast Multi-scale Method for Drawing Large Graphs," in *Proceedings of the 8th International Symposium on Graph Drawing*, ser. GD '00. London, UK, UK: Springer-Verlag, 2001, pp. 183–196.
- [7] R. Hadany and D. Harel, "A Multi-Scale Algorithm for Drawing Graphs Nicely," *Discrete Applied Mathematics*, vol. 113, no. 1, pp. 3–21, 2001, selected Papers: 12th Workshop on Graph-Theoretic Concepts in Computer Science.
- [8] C. Walshaw, "A Multilevel Algorithm for Force-Directed Graph Drawing," in *Proceedings of the 8th International Symposium on Graph Drawing*, ser. GD '00. London, UK, UK: Springer-Verlag, 2001, pp. 171–182.
- [9] Y. F. Hu, "Efficient and High Quality Force-directed Graph Drawing," *The Mathematica Journal*, vol. 10, pp. 37–71, 2005.
- [10] C. Mueller, D. Gregor, and A. Lumsdaine, "Distributed Force-Directed Graph Layout and Visualization," in *Proceedings of the 6th Eurographics Conference on Parallel Graphics and Visualization*, ser. EGPGV '06. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, 2006, pp. 83–90.
- [11] A. Tikhonova and K.-L. Ma, "A Scalable Parallel Force-Directed Graph Layout Algorithm," in *Proceedings of the 8th Eurographics Conference on Parallel Graphics and Visualization*, ser. EGPGV '08. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, 2008, pp. 25–32.
- [12] S. G. Kobourov, "Spring Embedders and Force Directed Graph Drawing Algorithms," *CoRR*, vol. abs/1201.3011, 2012.
- [13] S. Lloyd, "Least Squares Quantization in PCM," *IEEE Trans. Inf. Theor.*, vol. 28, no. 2, pp. 129–137, Sep. 2006.
- [14] G. Liotta and H. Meijer, "Voronoi Drawings of Trees," *Comput. Geom. Theory Appl.*, vol. 24, no. 3, pp. 147–178, Apr. 2003.
- [15] M. Balzer and O. Deussen, "Voronoi Treemaps," in *Proceedings of the 2005 IEEE Symposium on Information Visualization*, ser. INFOVIS '05. Washington, DC, USA: IEEE Computer Society, 2005, pp. 7–.
- [16] A. Nocaj and U. Brandes, "Computing Voronoi Treemaps: Faster, Simpler, and Resolution-independent," *Computer Graphics Forum*, vol. 31, no. 3pt1, pp. 855–864, 2012.
- [17] Linkurious, "Linkurious.js," 2016. [Online]. Available: <https://github.com/Linkurious/linkurious.js>
- [18] A. Jacomy and G. Plique, "Sigma.js," 2016. [Online]. Available: <https://github.com/jacomyal/sigma.js>
- [19] R. Hill, "Javascript-Voronoi," 2016. [Online]. Available: <https://github.com/gorhill/Javascript-Voronoi>
- [20] H. C. Purchase, "Metrics for Graph Drawing Aesthetics," *Journal of Visual Languages & Computing*, vol. 13, no. 5, pp. 501–516, 2002.
- [21] J. Leskovec, J. Kleinberg, and C. Faloutsos, "Graph Evolution: Densefication and Shrinking Diameters," *ArXiv Physics e-prints*, Mar. 2006.
- [22] J. Leskovec and A. Krevl, "SNAP Datasets: Stanford Large Network Dataset Collection," Jun. 2014. [Online]. Available: <http://snap.stanford.edu/data>
- [23] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 10, p. 10008, Oct. 2008.
- [24] Y. Liu, W. Wang, B. Lévy, F. Sun, D.-M. Yan, L. Lu, and C. Yang, "On Centroidal Voronoi Tessellation—Energy Smoothness and Fast Computation," *ACM Trans. Graph.*, vol. 28, no. 4, pp. 101:1–101:17, Sep. 2009.
- [25] G. Rong, Y. Liu, W. Wang, X. Yin, X. Gu, and X. Guo, "GPU-Assisted Computation of Centroidal Voronoi Tessellation," *IEEE Transactions on Visualization and Computer Graphics*, vol. 17, no. 3, pp. 345–356, Mar. 2011.

Fernando Wong Obtained his Masters and Doctorate degrees in Computer Science at the Graduate School of Information Science and Technology of the University of Tokyo, in 2010 and 2013 respectively, where he focused on data visualization and computer graphics research. He has been working as senior research engineer at Allm Inc. since 2013, engaging mostly into research in data visualization and machine learning for medical applications.

Horacio Sanson Obtained his Masters Degree at the Global Information and Telecommunication Institute of Waseda University in Tokyo, Japan, in 2005, where he researched distributed localization on emergency wireless networks. From there, he worked on several engineering positions until he became lead research engineer at Allm Inc. on 2014, where he spends his time applying engineering and machine learning to solve practical medical and health-care issues.