Visualizing Community Centric Network Layouts

Justin Fagnan, Osmar Zaïane, Randy Goebel
Department of Computing Science
University of Alberta, Edmonton, Canada
{fagnan, zaiane, rgoebel}@ualberta.ca

Abstract

We present our COMmunity Boundary (COMB) and COMmunity Circles (COMC) network layout algorithms that focus on revealing the structure of discovered communities and the relationships between these communities. We believe this information is vital when developing new community mining algorithms as it allows the viewer to more quickly assess the quality of a mining result without appealing to large tables of statistics. To implement our algorithms we have introduced numerous modifications to the existing Fruchterman-Reingold layout, including support for multi-sized vertices, removal of the bounding frame, introduction of circular bounding boxes, and a novel slotting system. Our evaluation argues that both COMB and COMC outperform existing alternatives in their ability to reveal community structure and emphasize inter-community relations.

Keywords—**community mining, visualization, network layout**

1 Introduction

On a daily basis we participate in a number of information networks that constrain everything from how proteins interact within our bodies, to whom we communicate with through the web. These networks can often be expressed as a collection of entities, represented as vertices, and the relationships between these entities, represented as edges. For example, we can build information networks relating human proteins, where relationships are formed if two proteins interact strongly with each other. Likewise, we can generate information networks for mobile phone companies, where each phone call forms an edge between two customers in the network.

A key idea for analyzing these networks is known as "community mining," and algorithms to detect them have received significant attention for a variety of possible applications, including the detection of protein interaction communities in Biology, crime factions in Criminology, friendship cliques in Sociology, and many more.

Despite significant community mining development, there has been little evaluation of these algorithms and few visualization methods that showcase either the structure of the communities or the relationships between them. Without these visuals clues it can be difficult for researchers to develop any intuition when evaluating their algorithms. Furthermore, while visualizations are beneficial for non-research users, the lack of evaluation methods means that the selection of community mining methods relies more on aethstestics than accuracy.

We address this issue by presenting two different layout algorithms. Our algorithms aim to generate aesthetically pleasing graph layouts which accurately highlight the structure of each community and the relationships between the communities. We believe our layouts allow a user to more easily discern the difference between a good community mining algorithm and a poor one, when compared to existing generalized layouts. In addition, our layouts more accurately depict the relationships between the communities, thus supporting more accurate inference by both researchers and users.

To accomplish these goals, we first propose modifications to the existing Fruchterman-Reingold (FR) algorithm [4], including support for vertices with a non-zero radius, and the removal of visualization boundaries. We then introduce our first community centric layout algorithm, called Community Boundaries (COMB), which uses a two-stage process to generate representative vertices, and exploits circular bounding boxes to ensure the communities remain intact and isolated. Subsequently, we present our second algorithm, named Community Circles (COMC), which uses a slotting system to efficiently showcase the relationships between each community.

Our contributions are as follows:

1. Adding support for non-zero radius vertices to the existing FR algorithm, and removing the bounding frame.

2. COMB, our two stage algorithm for generating an efficient community-centric layout that highlights the relationships between communities.

3. COMC, our algorithm which generates community centric layouts using a highly efficient slotting system to showcase the relationships between communities.

Below we briefly summarize related work in Section 2, followed by the introduction of some minor enhancements to the existing Fruchterman-Reingold algorithm in Section 3. In Sections 4 and 5 we describe our COMB and COMC algorithms respectively. Section 6 provides a preliminary evaluation of our algorithms, and Section 7 provides a summary.

2 Related Work

The basis of our research arises from the well-known Fruchterman-Reingold (FR) layout algorithm, which aims to produce layouts that are both aesthetically pleasing and contain uniform edge lengths. To accomplish this, the FR layout employs a force-directed model where each vertex in the graph both attracts its neighbours and repels all other vertices [4]. The strength of this attraction/repulsion is regulated by the current "temperature" of the system: high temperatures produce strong attraction/repulsion forces, and low temperatures produce weak forces. These forces determine how far away a vertex should move if it is repelled, or how close it should come if it is attracted.

The FR algorithm is an iterative algorithm, based on three main functions: the attraction function, the repulsion function, and the position calculation function. At each iteration the attraction is computed between each end point of an edge, and the repulsion between all pairs of vertices. These forces are summed by the position calculation function, and each vertex is moved so that it becomes closer to its neighbours and further away from its non-neighbours. The temperature is then reduced, and the next iteration starts; the algorithm completes when the temperature reaches zero or the number of iterations reaches a threshold. We note here that because FR operates on edge end points, it can support multi-graphs by accumulating the attraction for all of the shared edges. We also note that, although the temperature system described is comparable to simulated annealing algorithms, the layout itself does not do any hill-climbing because of the time complexity to reach the optimal solution. Instead, the authors assume that the discrete iterative movement of vertices will be enough to overcome any local optima.

Other force-directed layouts have also been proposed, including the well-known Kamada-Kawai (KK) layout [7]. The KK algorithm treats edges between vertices as springs, and uses Hooke's law to compute the attractive forces between neighbouring vertices. Their algorithm attempts to minimize the total tension in the system; while also finding an "ideal" distance between non-neighbouring vertices. We know of no conclusive evaluation that determines which of FR or KK produces "better" layouts. But we have chosen FR as a basis for our extensions because of its popularity.

Numerous other community or cluster based layout algorithms have been proposed, including the energy model proposed by Truong et al. [13], Frishman and Tal [3], and Balzer and Deussen [1]. Our proposed methods differ from previous approaches in that we do not require any user-defined parameters to produce our community layouts, nor do we focus on any specific aesthetic quality. Rather, we attempt to maximize the distinctive aspects, such as the inter-community qualities, intra-community qualities, and the network as a whole. We argue that this generalized approach produces more meaningful visualizations and, contrary to previous methods, we provide an evaluation to showcase this.

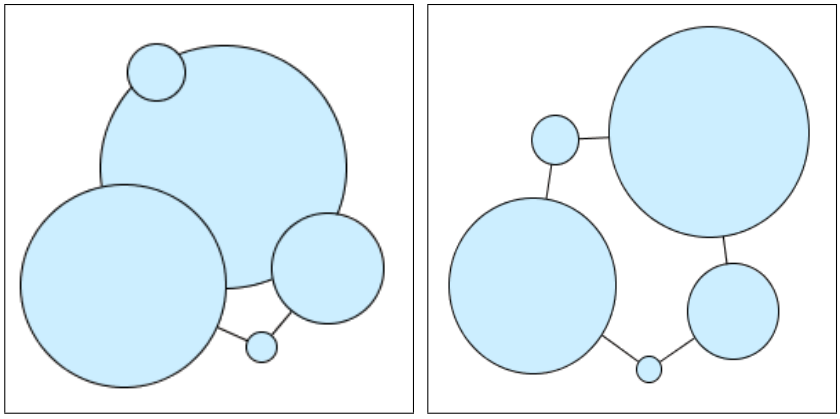
3 Fruchterman-Reingold Enhancements

3.1 Multi-Sized Vertices

Our layout algorithms first require us to introduce two basic modifications to the existing FR algorithm. The first modification addresses the practical need to visualize networks that contain vertices with heterogeneous sizes. Such networks are common in real-life scenarios where vertex size conveys some meaningful information to the viewer. Unfortunately, the existing FR algorithm assumes that all vertices are infinitesimally small points; thus the resulting layouts often conclude with large vertices overlapping or completely obscuring smaller vertices. An example of such a case is depicted in Figure 1(a). This overlap occurs because the repulsion function in the existing FR algorithm repels each pair of vertices by a force inversely proportional to the distance, d, between their centers. As d nears zero, the force becomes strong to prevent vertices from clumping. Yet this repulsion is insufficient when each vertex has a non-zero radius, because the vertices will be overlapping long before the distance between their centers nears zero. To address this, we apply an intuitive solution of measuring the distance, d, between the *edges* of the vertices, instead of the centers. In this way the value d accurately reflects the distance between vertices of any size, and the extremely strong repulsive forces will occur before the vertices begin to overlap. We should note that although our solution is independently conceived, it has been previously proposed by Harel and Koren [6]. More formally, we set the distance d equal to:

$$d = \max(dist(u, v) - (radius(v) + radius(u)), \epsilon)$$

Where $dist(u, v)$ is the Euclidean distance between the centers of vertex u and vertex v, and $radius(x)$ is the radius of the vertex x. We include ϵ as a small positive con-



(a) Original FR algorithm. (b) FR with support for sized vertices.

Figure 1: An example of our modification to support sized vertices in the FR algorithm.

stant to ensure that overlapping vertices do not result in a negative distance, d. The resulting layout is shown in Figure 1(b).

3.2 Boundary-Free Layout

Our second modification removers the FR algorithm's requirement for a user-defined frame or bounding box for the resulting layout. The FR authors argued that a layout is aesthetically pleasing when it conforms to a frame and is distributed evenly throughout the frame [4]. We, however, feel that this requirement is largely unintuitive. We do not often force a picture to fit a specific picture frame, but rather seek a frame that is large enough to contain our picture.

In removing this FR algorithm constraint, we redefine two of the constants in the existing FR algorithm: the force constant, k, and the starting temperature, $temp$. These constants were previously based on the bounding box width and height, and strongly influenced the distance a vertex is able to move in a single iteration. We provide the following alternative definitions:

$$k = \sqrt{(|V| + |E|)}$$
$$temp = |V| + |E|$$

where $|V|$ is the number of vertices in the multigraph, and $|E|$ is the number of edges. Our choice is based on our intuition about how much movement should occur relative to the size of the network. Large networks require larger vertex movements in order to escape a local minima and thus their k and $temp$ values should be higher than smaller networks. We have chosen a square root function to ensure that very large networks do not result in extreme movements that end up accomplishing little else other than wasting an iteration.

4 Community Boundaries

Here we present our first layout algorithm, COMB, which works in two stages. First, we compute the boundary size for each community and then determine its final position in the layout. In the second stage, we place each vertex within its community boundary and attempt to minimize the inter-community edge lengths.

Our algorithm operates on a multigraph $G = \{V, E, C\}$ where V is the set of vertices, $V = \{v_1, v_2, v_3...\}$, E is the set of edges, $E = \{e_1, e_2, e_3...\}$, and C encodes the computed communities, $C = \{c_1, c_2, c_3...\}$. We define each community $c_x = \{V_{c_x}, E_{in_x}, E_{out_x}\}$ where V_{c_x} is the set of vertices in the community, E_{in_x} is the set of edges between two vertices within the community, and E_{out_x} is the set of edges leading out of the community.

4.1 Representative Vertices

Initially, our algorithm identifies a single representative vertex for each community in the network. These representative vertices will ultimately determine the position of the communities in the final layout. Therefore each of these representatives must be structurally equivalent to the community they represent, in terms of both visual size on the screen and connections to other vertices.

To address connection equivalence, we generate a set of representative vertices, $R = \{r_{c1}, r_{c2}, ..., r_{cn}\}$, such that the set of edges that each r_{cx} participates in, $E_{r_{cx}}$, is exactly equal to E_{out_x}, which is the set of edges its representative community participates in. Since it is likely that each pair of representative vertices will be connected by more than one edge, we need a multigraph structure.

To compute the on-screen size of a community we employ one of two techniques. The first technique is the obvious solution of generating a separate layout for each community using the FR algorithm, then computing the convex hull of the layout [4]. This gives us a fairly accurate esti-

mate of how much visual space the community will consume in the final layout. However, it is costly to run the FR algorithm ($O(|V^2| + |E|)$) for each community in the network. An alternative is to estimate the required visual space using only the number of internal edges ($|E_{in_x}|$), vertices ($|V_{cx}|$) within each community. We have defined our estimate function as:

$$radius = (\max(10, \sqrt{|V_{cx}|}) + \max(10, \sqrt{|E_{in_x}|})) * lm$$

Where lm is a constant derived from the average radius of each vertex. We employ the max functions to ensure that very small communities are allotted some meaningful visual space. Note that any other estimate function could also be used in our algorithm; we simply chose one that makes some intuitive sense and has been satisfactory in our experiments. By using either the estimate or the separate layout technique, we can then set the radius of each representative vertex, r_{cx}, to be equal to the computed radius of the bounding circle for community x. An example of generating representative vertices from a set of communities is presented in Figure 2.

4.2 Initial Layout

After generating the representative vertices, we can produce an abstraction of the network, $G_{abstract} = \{R + V_{no-comm}, E\}$, where R is our set of representatives and $V_{no-comm}$ is the set of vertices that do not belong to any community and thus are not captured by the representatives (an example of $G_{abstract}$ can be seen in Figure 2(b)). We then feed this abstract network into our FR algorithm that has been modified to support vertex sizes.

The resulting layout contains representative vertices placed at their desired positions, given the goal of uniform edge length. These positions are recorded and will be used to determine where each community should reside in the final layout. Note that we do not record the position of the vertices in $V_{no-comm}$. These vertices were only added to the abstraction to ensure that their edges are considered when finding an optimal position for the representatives.

4.3 Vertex Placement

In the second stage of COMB we focus on refining our initial placement by setting the location of each vertex in the network. We begin this stage with the initial layout generated in stage 1, which contains the representative vertices. We then replace each representative vertex with a bounding circle, centered at the same location that the vertex once occupied. These bounding circles will ensure that the final layout has clearly defined boundaries for each community. As such, each circle must have a radius equal to that of the representative vertex it replaced.

Once the bounding circles are appropriately sized we begin inserting the vertices from the original network. Each vertex is placed at a uniformly selected random position within the circle that represents its community. After the placement is complete we have an initial layout where all vertices belonging to the same community are contained within the same bounding circle. When we feed this initial placement into our modified FR algorithm we expect the vertices to move around within their circle, but they can not leave the circle. This limited freedom of movement should allow those vertices with many connections to other communities to migrate towards the perimeter of their respective circle. In the following section we define these bounding circles.

4.4 Implementing Bounding Circles

To enable bounding circles we have modified the FR position calculation function so that each vertex is checked against the perimeter of its bounding circle in each iteration. We confirm whether the vertex is outside the perimeter by calculating the distance between the vertex and the center of the circle. Any vertex with a distance greater than the radius is outside of the perimeter, and needs to be relocated inside the perimeter. Before doing so, we record the vertex's angle to the center of the circle. This angle should point towards the "optimal" position (as determined by the FR algorithm) that the vertex was trying to reach, and thus it may be the optimal angle within the bounding circle as well.

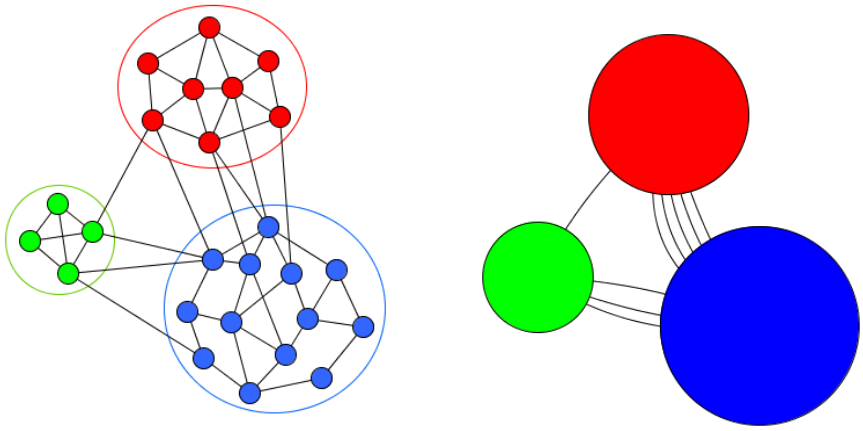
To achieve this, we first compute the angle between the out-of-bounds vertex and the center of the circle:

$$angle(v, c) = atan2(c.y - v.y, v.x - c.x)$$

Where v is the vertex, c is the bounding circle that contains the vertex, and x/y refers to the x and y coordinates of a point. $c.x$, for example, would indicate the x coordinate for the centre point of the circle c.

Once the angle is computed, we can move the vertex within the circle, But how far inwards? Our early experimentation showed that moving the vertex exactly to the perimeter produced poor layouts, as the perimeters of the circles became crowded and obscured the structure of the community itself. To combat this problem, each offending vertex is moved to a randomly selected position between the perimeter and the center of the circle, while keeping the same angle. Our choice of randomness follows a uniform distribution.

However, randomness alone is not satisfactory because a vertex may be incorrectly moved to the center of the circle on the very last iteration, even though it just barely slipped outside of the bounding circle. Therefore we also take into account the temperature of the system: when the temperature is near its maximum, the offending vertices may be moved all the way to the center of the circle. As



(a) Three communities (b) The generated Representative Vertices

Figure 2: An example of generating Representative Vertices.

the temperature cools, the movements do not stray far from the perimeter. We capture this intuition in the following formulae:

$$mlen(c) = c.radius * \left(1 - \left(rand() * \frac{curTemp}{maxTemp}\right)\right)$$
$$length = mlen(c)$$
$$v.x = (\cos(angle(v,c)) * length) + c.x$$
$$v.y = (\sin(angle(v,c)) * length) + c.y$$

Where c is the bounding circle for the vertex v, $angle()$ is previously defined, $rand()$ returns a uniformly random number [0,1], $curTemp$ is the current temperature, and $maxTemp$ is the initial temperature.

As the system cools, $curTemp$ decreases, causing $length$ to increase, which places the vertices (on average) further away from the center of the circle and closer to its perimeter. This ensures that vertices which barely slip out of the circle in the final iterations of the algorithm are still able to remain near the perimeter.

4.5 Final Layout

Now that the bounding circles are defined we merely provide the entire network, G, to our modified FR algorithm. We then feed the initial placement of the vertices, and the definitions of each bounding circle, into the algorithm. The resulting layout contains all of the features of the original FR algorithm along with a well-defined visual boundary around each community. Visualizations are provided in the Evaluation section.

5 Community Circles

The layouts produced by COMB contain clearly defined boundaries and reveal both the internal structure and relationships of the communities. Highlighting this internal structure, however, has an efficiency cost as the layout algorithm needs to retain all of the functions from the original FR algorithm, resulting in an $O(|V|^2 + |E|)$ time complexity. This is comparable with existing layout algorithms, but it is not fast enough to be considered a truly interactive visualization. Thus we propose our COMmunity Circles (COMC) algorithm, which maximizes efficiency by using circular layouts to showcase only the community relationships. Our goal with COMC is to represent each community as a circle and place the vertices on the perimeters of each circle such that inter-community edge length is minimized. This algorithm is similar to COMB, in that it is based on FR, and shares many of the same steps, including the generation of representative vertices and the resulting vertex replacement.

5.1 Sizing Representative Vertices

Unlike in COMB, we do not need to scale the size of the representative vertices according to the number of edges within the community, as these edges are essentially ignored by the COMC layout. Instead, we need to ensure that the perimeter of each representative is large enough so that we can place each vertex on it without overlap. Thus we modify the radius function:

$$circumference = (avgsize(V_{cx}) + padding) * |V_{cx}|$$
$$radius = \frac{circumference}{2\pi}$$

where $avgsize()$ is the average radius of all the ver-

tices in V_{cx}, and $padding$ is a user-defined constant that describes how much space there is between each vertex on the circle. Our preferred setting $padding = 15$.

5.2 Perimeter Slots

Once the representatives are generated and placed using COMB, we can record the perimeter and location of each representative and use this information to define a new data structure. This structure contains both the center and radius of the circle along with an array of "slots," one for each vertex in the community. During the course of the COMC algorithm, each vertex is assigned to the closest slot on its respective community circle, according to the Euclidean distance between the circle and the current position of the vertex. If the closest slot is already occupied by another vertex, we compute the total length of all inter-community edges for both vertices. The vertex with the smallest total edge length gets the slot, and the other vertex must move to the second closest slot. If that slot is also occupied, then the procedure repeats itself until there is a suitable unoccupied slot. We can guarantee that this process terminates because the set of inter-community edge lengths has a total order.

Before assigning the slots, we determine the position of each vertex by leveraging the attraction and repulsion functions of the FR algorithm. In particular, we only compute the attraction between two vertices if they belong to different communities. This is because we do not need to consider the attraction between members of the same community; they are all tied to the perimeter of the circle and cannot move closer to each other. In addition, we do not need to compute the repulsion function for any vertex that belongs to a community. We can avoid this step because a pair of vertices within a community cannot possibly get further away than the diameter of circle and we not interested in showcasing the internal structure of each community.

The attraction function sets the position of each vertex by drawing it closer to its neighbours in other communities. Once this position is set, each vertex is assigned to the nearest slot on its community circle as described above. At the end of each iteration, the vertices are moved to the position dictated by their respective slot, and then the slots are cleared for the next iteration. The algorithm continues until the system has cooled according to a cooling function specified in the FR algorithm [4].

5.3 Efficiency Gains

By removing the repulsion function and limiting the attraction function, we have greatly reduced the time complexity of the algorithm. In each iteration, we need only loop over the set of inter-community edges to compute the attraction function and then assign the slots. This reduces the worst-case time complexity from $O(|V^2| + |E|)$ down to $O(|E_{out}| + |V| * avg(|C|))$, where $avg(|C|)$ is the average number of vertices in each community. In practice, the slot assignments tend to stabilize after the first few iterations of the algorithm, and thus the time complexity is dominated by $|E_{out}|$. A sample visualization of COMC is provided in the Evaluation section.

6 Evaluation

6.1 Visual Juxtaposition

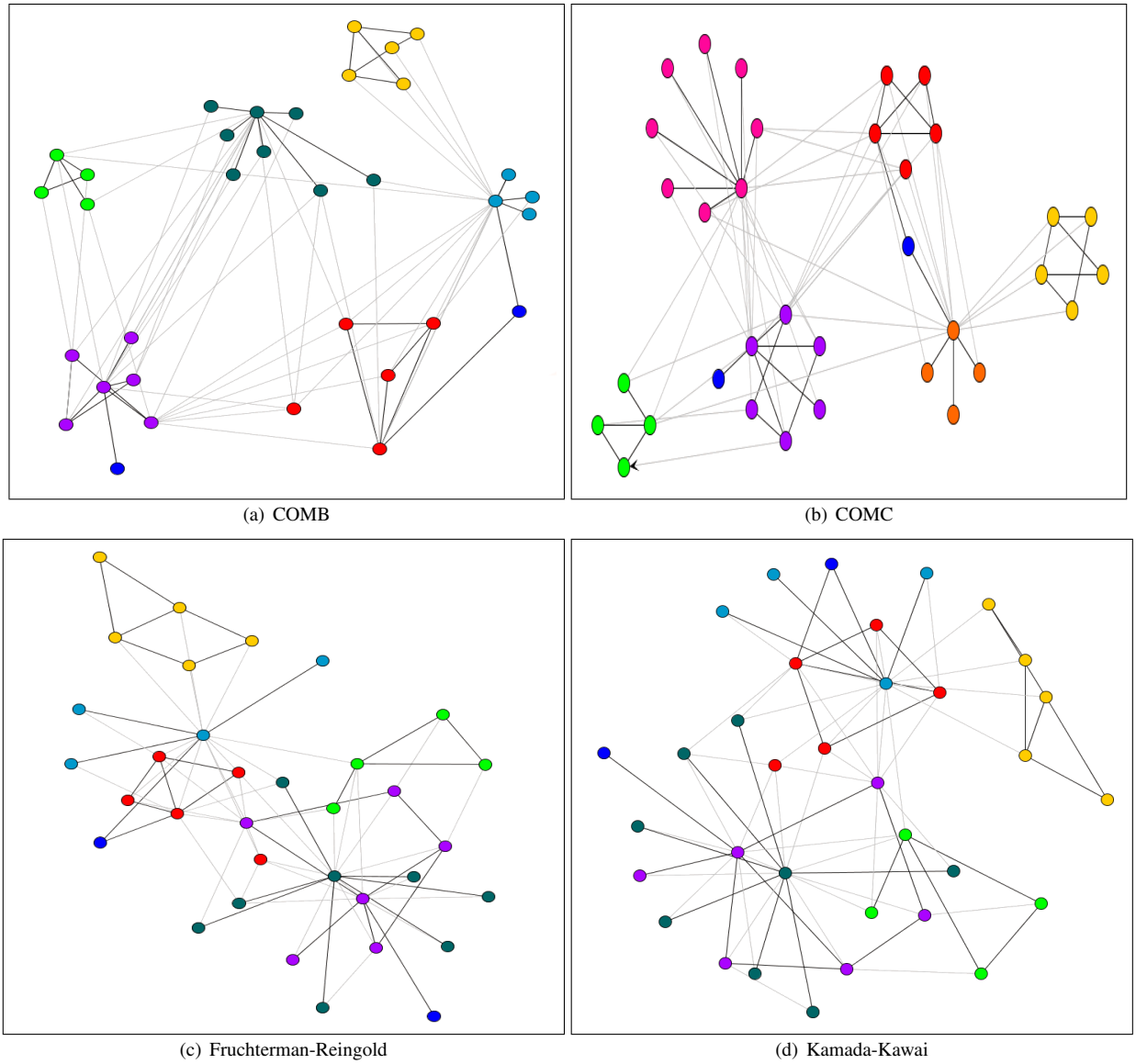
Research in visualization evaluation (e.g., [12, 11]) suggest an evolution towards the design of cognitive experiments to establish measures of visualization effectiveness. But a useful precursor to such formal evaluation is to provide informal arguments of visualization method preferences by comparing alternative renderings of the same kind of base data. In this way we at least begin the development of intuitive measures that provide the basis for visualization method preference. We have adopted this approach.

In our comparison of alternative methods for community clustering visualization, we are unable to include any of the related work algorithms because we could not find implementations and the examples provided in the respective papers were trivial at best.

In Figures 3, 4, and 5 we present visualizations for a variety of well-known social networks, including the NCAA Football network [5], Zachary's Karate Club [14], and the Political Books network[8]. For each network we have run an existing Community Mining algorithm, such as Fast Modularity [2], and generate a visualization using COMB, COMC, the Kamada Kawai (KK) algorithm [7], and the general Fruchterman-Reingold algorithm [4]. We note that, although the KK and FR algorithms are not specifically tailored for communities, they are nonetheless the most popular methods used to visualize community mining results. To generate the visualizations we have used the FR and KK implementations provided by the JUNG framework [9]. Note that the color of a vertex specifies the community it belongs to.

For the rather trivial Karate Club network depicted in Figure 3 we can already see a clear distinction between our proposed algorithms and the generalized layouts. In both the COMB and COMC layouts we can easily see the relationships between the communities and, in the case of COMB, the structure of each community. This insight would allow the viewer to recommend that perhaps the yellow and teal communities should be merged. Similar observations are much harder to make in the generalized layouts because it is difficult to tell how each community is related. So in this case, an informal notion of evaluation notes that there are emergent inferential properties of alternatives, even without formal evaluation.

For the Political Books network shown in Figure 4 we can see that both the COMB and COMC layouts highlight



(a) COMB (b) COMC

(c) Fruchterman-Reingold (d) Kamada-Kawai

Figure 3: Zachary's Karate Club Network.

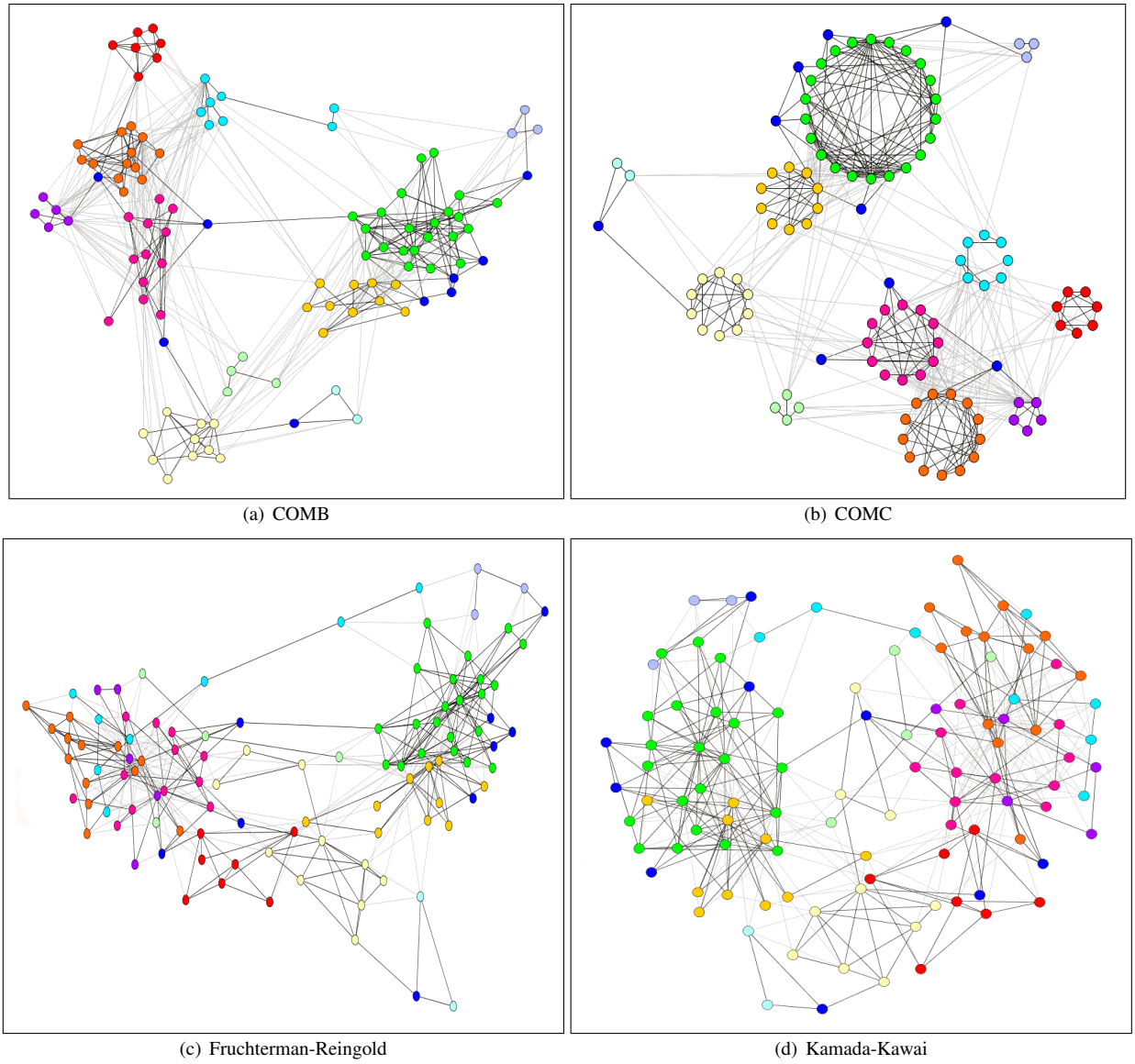
the tight coupling between the orange, pink, and purple communities, whereas one can only assume such a coupling exists in the FR and KK based layouts.

Finally, in the NCAA Football network, shown in Figure 5, we can see clear community boundaries and the interaction between these communities in both the COMB and COMC layouts. Furthermore, both the FR and COMC layouts reveal that each of the communities contain numerous outliers (the single edge vertices), indicating that perhaps a community mining algorithm with support for outliers should be used instead. Unfortunately, the FR layout provides no other information as the communities seem to be completely overlapping each other, making it difficult to determine the structure of any community. The same can be said for the KK layout.

6.2 Efficiency

In addition to these informal visual evaluation distinctions, we also present a comparison on the efficiency of the algorithms. The COMB, FR, and KK algorithms employ a force-based approach that requires computing a force between every pair of vertices, and the end points of each edge, in each iteration. Thus the time complexity of each iteration can be thought of as $\Theta(|V|^2 + |E|)$; our modifications to the FR algorithm do not add any significant time complexity.

Computing the final time complexity requires bound-



(a) COMB (b) COMC

(c) Fruchterman-Reingold (d) Kamada-Kawai

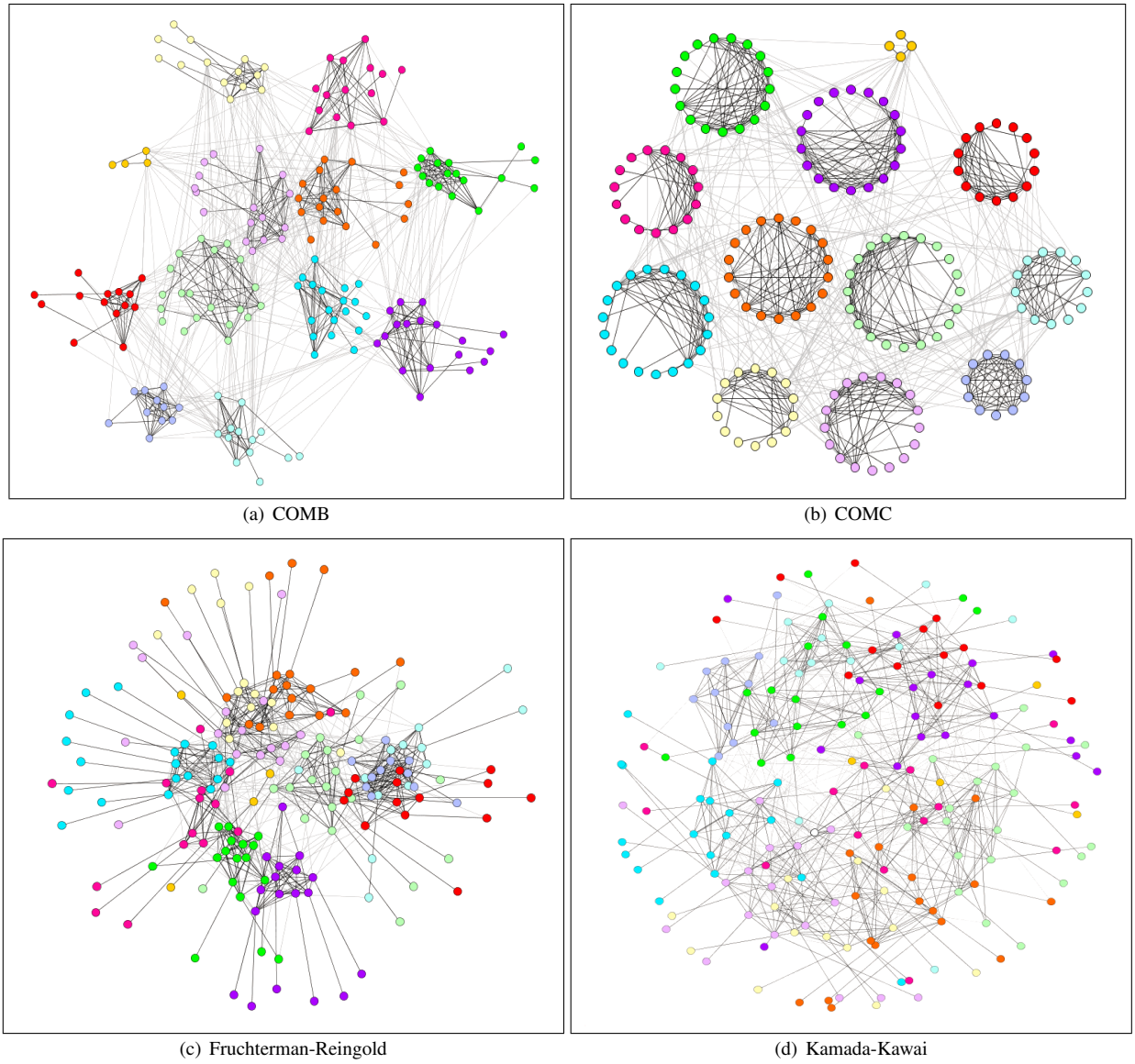
Figure 4: Political Books Network.

ing the possible number iterations; yet this is a notoriously difficult question in systems with cooling schedules. To avoid this issue, previous authors have simply set a limit to the number of iterations, say 100. We have adopted this approach and determined that the final time complexity of COMB, FR, and KK is $O(|V^2| + |E|)$. The COMC algorithm does not need to compute the pairwise forces and thus its worse case time complexity is $O(|E_{out}| + |V| * avg(|C|))$.

A list of the running times for each algorithm, computed on an Intel i7-2540m with 8GB of RAM, is shown in Table 1. We can see that COMB is equivalent to the existing FR algorithm in terms of efficiency, and that COMC is significantly faster than the other layouts. We believe this efficiency allows one to use the layout in an interactive setting, such as displaying the evolution of communities in dynamic networks. We should note that the Kamada-Kawai algorithm appears to be much slower because it spends considerable time making minor adjustments to the layout.

7 Summary and Future Work

We have presented two novel layout algorithms, COMB and COMC, that aim to generate aesthetically pleasing network layouts while highlighting both the structure of each community and the relationships between the communi-



(a) COMB (b) COMC

(c) Fruchterman-Reingold (d) Kamada-Kawai

Figure 5: NCAA Football Network.

Network	V	E	COMB	COMC	FR	KK
Karate Club	34	77	143 ms	85 ms	130 ms	110 ms
Politcal Books	105	441	7567 ms	375 ms	7181 ms	19752 ms
NCAA Football	180	787	13585 ms	411 ms	14258 ms	22530 ms

Table 1: Runtimes of each layout algorithm, averaged over 10 runs.

ties. To accomplish this goal we modified the existing FR algorithm by removing the need for bounding borders, adding support for vertices with non-zero size, and either enabling the definition of bounding circles within the layout or perimeter slotting[4].

Together, these modifications allowed us to create distinctive visual borders around each community (COMB), and greatly improve the efficiency of current visualization methods (COMC) while still generating an aesthetically pleasing layout. Our evaluation argues that both COMB and COMC offer more insight into the structural components of each community and the relationships between the

communities when compared to the existing techniques. We hope that our visualizations allow researchers and analysts to more quickly identify communities of interest without needing to inspect tables of statistics.

In our future work, we intend to extend COMB to support other popular layout algorithms, including the Kamada-Kawai [7] and ISOM [10] layouts. We may also investigate the inclusion of polygonal bounding boxes which should allow for a more natural layout of each community. Unfortunately, the operation to determine if a vertex is outside of a polygon is considerably more expensive than for a circle. A naive implementation of polygon bounding boxes may make the algorithm prohibitively expensive for larger networks.

We are also unsatisfied with the current methods of evaluating visualization algorithms; it is clear that more objective metrics or survey methods are required to determine how a "good" community layout would differ from a "bad" one. It is a daunting task to present a new layout algorithm without further development of a methodology to compare visualization results.

References

[1] M Balzer and O Deussen. Level-of-detail visualization of clustered graph layouts. In *APVIS'07*, pages 133–140, 2007.

[2] A Clauset, M.E.J. Newman, and C. Moore. Finding community structure in very large networks. *Phys. Rev. E*, 70, 2004.

[3] Y Frishman and A Tal. Dynamic drawing of clustered graphs. In *Proceedings of the IEEE Symposium on Information Visualization*, pages 191–198, 2004.

[4] T. M. J. Fruchterman and E. M. Reingold. Graph drawing by force-directed placement. *Software: Practice and Experience*, 21:1129–1164, 1991.

[5] M Girvan and M. E. J. Newman. *Proc. Natl. Acad. Sci.*, 99:7821–7826, 2002.

[6] D Harel and Y Koren. Drawing graphs with non-uniform vertices. In *Proceedings of the Working Conference on Advanced Visual Interfaces*, AVI '02, pages 157–166, 2002.

[7] T Kamada and Si Kawai. An algorithm for drawing general undirected graphs. *Information Processing Letters*, 31(1):7–15, 1989.

[8] V. Krebs. http://www.orgnet.com, January 2012.

[9] J. Madadhain, D. Fisher, P. Smyth, S. White, and Y.B. Boey. Analysis and visualization of network data using jung. *Journal of Statistical Software*, 10:1–35, 2005.

[10] B Meyer. Self-organizing graphs a neural network perspective of graph layout. In *Graph Drawing*, volume 1547 of *Lecture Notes in Computer Science*, pages 246–262. 1998.

[11] Chris North. Toward measuring visualization insight. *IEEE Comput. Graph. Appl.*, 26(3):6–9, 2006.

[12] Catherine Plaisant. The challenge of information visualization evaluation. In *Proceedings of the working conference on Advanced visual interfaces*, AVI '04, pages 109–116. ACM, 2004.

[13] Q.D Truong, T. Dkaki, and P.J. Charrel. An energy model for the drawing of clustered graphs. In *Proceedings of Veme colloque international VSST*, 2007.

[14] W. W. Zachary. An information flow model for conflict and fission in small groups. *Journal of Anthropological Research*, 33:452–473, 1977.