Graphemes: self-organizing shape-based clustered structures for network visualisations

Ross Shannon
Systems Research Group,
School of Computer Science &
Informatics,
UCD Dublin,
Ireland
ross.shannon@ucd.ie

Aaron Quigley
HITLab Australia,
School of Computing and
Information Systems,
University of Tasmania,
Australia
aaron.quigley@utas.edu.au

Paddy Nixon
Systems Research Group,
School of Computer Science &
Informatics,
UCD Dublin,
Ireland
paddy.nixon@ucd.ie

CHI 2010, April 10–15, 2010, Atlanta,
Georgia, USA.
ACM 978-1-60558-930-5/10/04.

Abstract
Network visualisations use clustering approaches to simplify the presentation of complex graph structures. We present a novel application of clustering algorithms, which controls the visual arrangement of the vertices in a cluster to explicitly encode information about that cluster. Our technique arranges parts of the graph into symbolic shapes, depending on the relative size of each cluster. Early results suggest that this layout augmentation helps viewers make sense of a graph's scale and number of elements, while facilitating recall of graph features, and increasing stability in dynamic graph scenarios.

Keywords
Dynamic graphs, graph drawing, visual memory.

ACM Classification Keywords
H.5.0. Information interfaces and presentation (e.g., HCI): General.

General Terms
Human Factors, Theory.

Introduction
Visually presenting complex network structures is a challenging area, and has resulted in a whole community of practitioners focused on graph layout algorithms [9]. Of the many successful approaches that have emerged, two in particular have many varied uses: force-directed layout algorithms [5], and graph clustering [13].

Force-directed layouts apply a set of attractive and repulsive forces between the vertices and edges of a graph, naturally generating a layout over time as a stochastic process models the interacting forces. Dynamically-changing graphs introduce challenges for the viewer in effectively maintaining their internal "mental model" of the structure under study [1], but force-directed approaches aid the viewer by smoothly interpolating between iterative versions of the layout, helping to preserve the mental model [7].

Clustering techniques have been used in machine learning to find commonality in large corpuses of data. In the visualisation of tree and network structures, clustering is intuitively applied to groups of vertices using graph-theoretic properties such as shortest path algorithms [3]. Vertices can be visually drawn close together or otherwise associated in the view, or, if the view is crowded or display space is limited, some vertices may be *elided* (removed from view) to simplify the drawing and call attention to key structural properties.

Visual clustering and elision in 2D or 3D will often take the form of multiple vertices in the original network

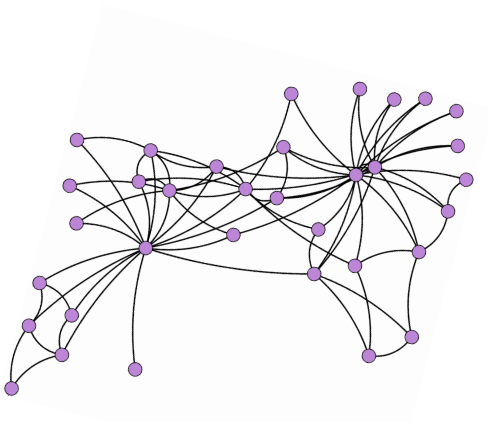


Figure 1. Even relatively small graphs can be challenging to analyse, navigate and manipulate. This graph of a social network has 34 vertices and 76 edges [14].

being represented as one, generally larger vertex in the resulting information space. The visualisation designer sacrifices representational fidelity for increased effective readability of the data that *is* presented. Further levels of visual abstraction may be attained if the desired complexity of the graph drawing has not been reached.

While the interests of the graph drawing community have become focused on ever more large and complex graph structures, it is important to remember that the comparatively small graphs that were visualised at the inception of the discipline are still very much relevant today. These graphs represent networks on a human scale: though the overall size of an online social network may be in the millions, the part of it that is relevant to a single user—that is, their own network of friends and acquaintances—is made up of roughly 150 people [4]. Graphs of only a few dozen vertices and edges like Figure 1 are found in computer interfaces, videogame menus and information software. There still exist opportunities to improve understanding, analysis and recall of graphs at this scale.

This article presents a novel technique applicable to small or clustered graphs which is designed to improve recall and understanding by exploiting the human brain's affinity for visual pattern-matching. The following section introduces the theory that has led to this approach. Following this are some implementation details and a discussion of early findings.

Symbolic Shapes

A "grapheme", in written languages, is a single character such as an alphabetic letter or number. Graphemes are the fundamental building blocks of meaning, which are combined together to form more complex models.

Each of us can distinguish a large set of visual forms and shapes quite easily, from the character sets used in languages we are each familiar with, to the almost universal set of shapes we know as squares, circles, triangles and so on.

Memory-supporting techniques such as the visual *mind map* seen in Figure 2 and utilised by students learning about a topic, recommend the use of color, position, flowing lines and iconography to enhance ease of recall [2]. The brain responds strongly to spatial layout in particular, which has significant primacy in recall performance [11].

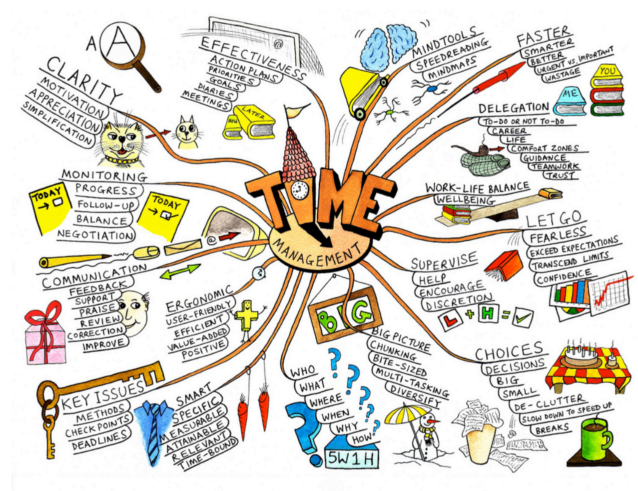
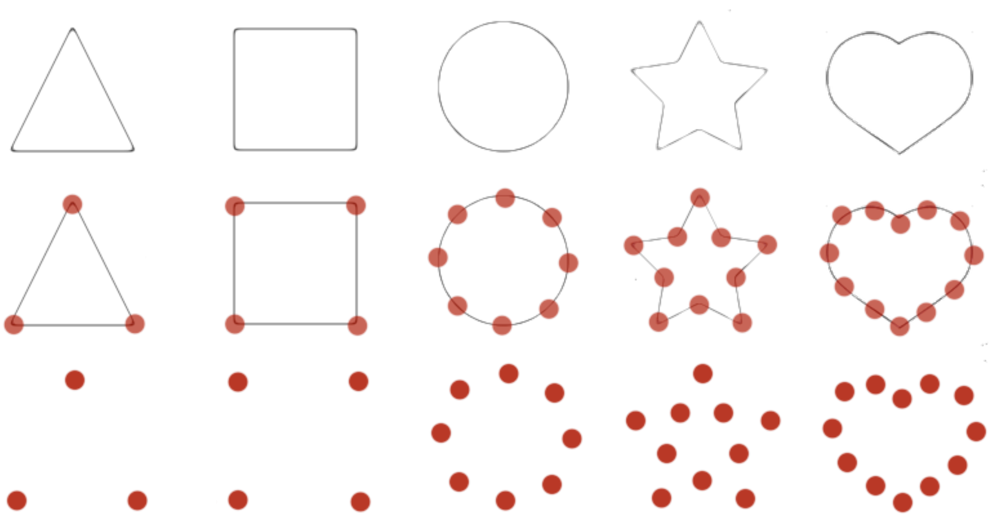


Figure 2. A mind map for "Time management", showing rich use of icons, color and spatial layout, all of which make the salient parts of the drawing easier to recall after viewing.[1]

Figure 3. A range of representations of common geometric shapes. The top row shows a triangle, square, circle, star and heart respectively from left-to-right. The second row shows the key vertex positions making up each of these shapes in our system. Finally, the third line shows these shapes in their vertex representation with no guidelines. Each shape remains distinguishable even at this low resolution.

Our *grapheme* technique is the combination of some of the benefits of both force-directed layouts and clustering algorithms to augment the layout that results from the layout algorithm alone. We apply customised sub-layouts to certain parts of the graph, arranging them into geometric shapes, converting unstructured clusters of vertices into nominal landmarks in the overall structure.

The graph is first partitioned into subsets using an edge-betweenness metric for detecting communities in complex networks [8]. This results in a list of subsets of the graph. In a person's social network these might be work colleagues, family, sporting friends—communities that are intra-connected but have few links between these subsets.

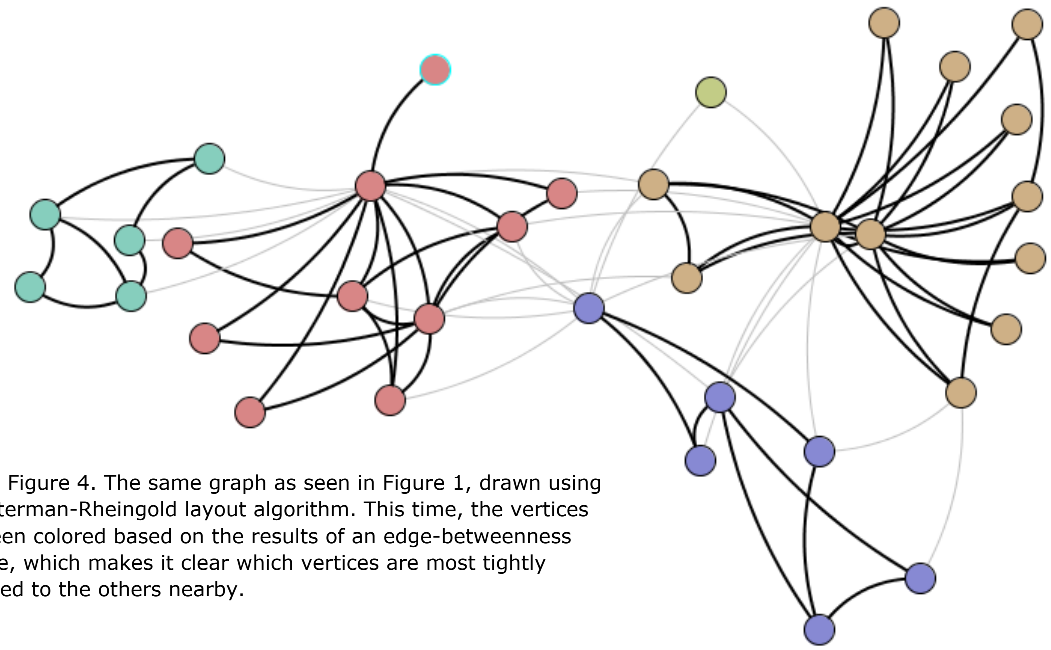


Next, we map the size of the clusters that have been detected to a set of five primitive geometric shapes (listed in Figure 3). Depending on the relative number of vertices in each subset, an appropriate shape from the range is chosen.

Shape Template Wireframes
Though further common shapes are available for use—such as arrows and more eccentric ellipses, hexagons etc.—at low resolutions they become difficult to distinguish. The five shapes we currently use were chosen for their familiarity (in fact, a "circle layout" is already a common graph layout scheme), and their horizontal symmetry; but also because they each can be easily broken down into a set of key vertex positions. If we have more vertices than we need key points, the shapes simply increase in fidelity as we position further vertices on the lines between two required key positions (this effect can be seen in the blue rectilinear cluster in Figure 4).

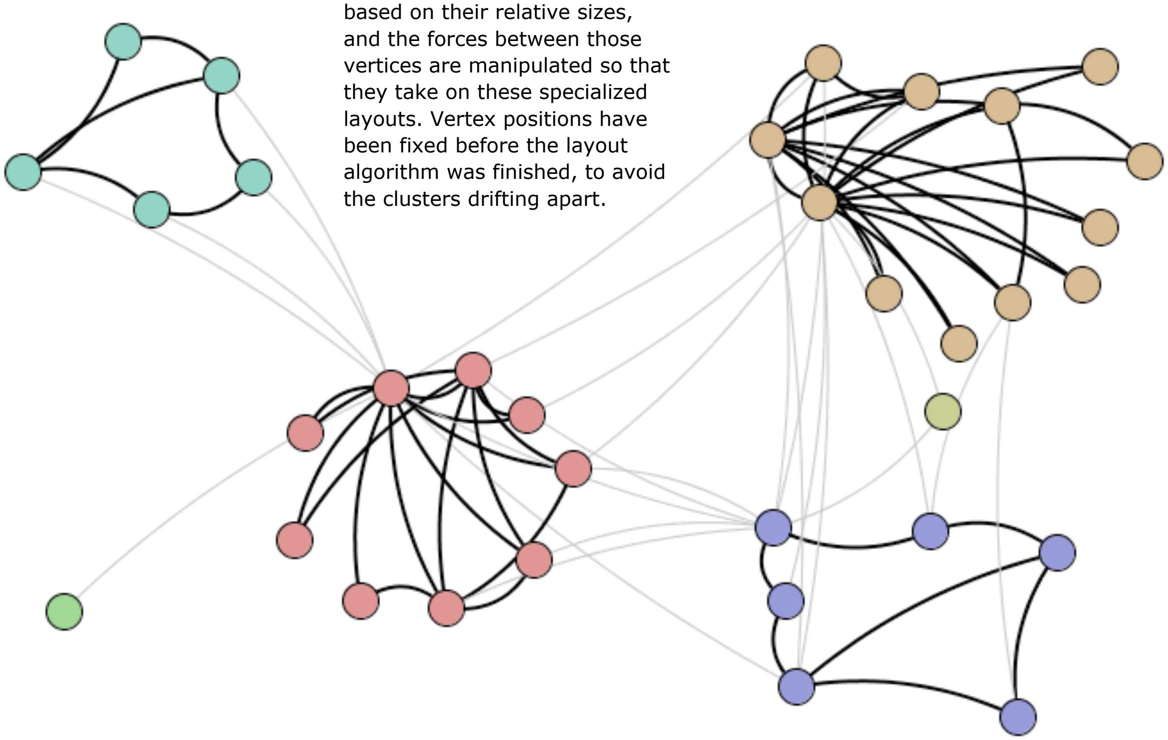
The complexity and number of vertices needed to represent the shapes increases as we go left to right in Figure 3, from the triangle, which needs only three points at minimum, through the square, circle, five-point star and heart, which needs twelve to be reasonably represented. This gives us a range to work with so that clusters of different sizes will be drawn using different shape templates.

The benefit of primitive shapes is, we hypothesise, that the resulting arrangements can be remembered symbolically, and the relative positions are more easily recalled, similar to how a star-gazer can remember the relative positions of abstract patterns in the constellations of the night sky.



(above) Figure 4. The same graph as seen in Figure 1, drawn using a Fruchterman-Rheingold layout algorithm. This time, the vertices have been colored based on the results of an edge-betweenness measure, which makes it clear which vertices are most tightly connected to the others nearby.

(below) Figure 5. After the partitioning process, each subset is assigned a shape template based on their relative sizes, and the forces between those vertices are manipulated so that they take on these specialized layouts. Vertex positions have been fixed before the layout algorithm was finished, to avoid the clusters drifting apart.



Implementation
A prototype implementation has been developed using JUNG, a Java tool for processing and visualising network data [12]. The figures in this article have used the Zachary's karate club data set which is well-known in social sciences [14]. Figure 4 shows the graph drawn with the built-in Fruchterman-Rheingold layout algorithm and colored to show the four main clusters (green, red, blue and orange). One yellow vertex is not part of any cluster.

Figure 5 shows the final layout. First, the graph is modified to use a force-directed layout, with starting positions derived from the vertices' positions from Figure 4. When assigning vertices to a shape, the key positions (from the shape template) are filled first. Vertices are supplied with forces which will eventually move them into position (for example, the vertices on opposite corners of the square template will repel each other strongly, but are relatively attracted to the other two corners). If any vertices are left over, they are positioned on a line between two key positions. The initial implementation of this issue places the additional vertex between two random key positions.

The library gives us the ability to selectively loosen and lengthen the edges between vertices in different subsets programmatically (e.g., the edge between a vertex in the blue subset and one in the orange subset in Figure 4). These inter-subset edges are rendered more lightly in both figures. At the same time we can stiffen the springs between vertices that are in the same subset, fixing them in place relative to each other. This gives an effect not dissimilar to the result of using virtual vertices to exert forces in the layout [6] (though that approach is more robust). Slightly different parameters for the clustering algorithm have resulted in a second unclustered vertex in this figure.

Some artefacts can be noted in the figures. In particular, the triangle shape which should be in effect in the light green subset in the top left is misshapen due to too much force being applied between two vertices at one of its corners. The heart shape is also not ideal.

Discussion

This technique was developed with the aim of aiding two cognitive processes that are known to be put to the test when analysing a complex network structure: the appreciation of the size of a graph and an estimation of number of vertices and number of clusters; and the memory of the arrangement of the graph in general, and the relative positions of clusters of vertices in particular, once the graph can no longer be consulted.

The five shape templates can be arranged ordinally in terms of how many nodes they are made up of. The system was designed this way with a view to helping viewers make better estimates of the size of the clusters, and of the whole graph. If our hypothesis is correct, shapes provide a way of *counting without counting*, using only pattern analysis to estimate figures without counting any individual vertices.

Since the human short term memory has only a finite capacity for storing stimuli [10], replacing amorphous clouds of vertices with organised arrangements of vertices that can be remembered as a single symbolic unit would seem to be a promising approach.

In the case of dynamic graphs, this technique can be used to call attention to significant events, like vertices being removed from a cluster, since the algorithm can be run again, resulting in one shape transforming into another after a significant change to the graph structure. Likewise, stable areas of the graph structure remain stable in the view.

Outstanding Issues

The interplay between laying out clusters of vertices in these shapes while they are also representing other multivariate properties (via their color or size, for example) is still unknown.

It is unclear if this type of approach to adding landmarks to a graph can scale up to hundreds or thousands of nodes. Nested layouts, with different layout algorithms being applied at different levels of abstraction could provide a way to use these techniques at low levels of a much larger network.

The technique does not perform well in graphs with very sparse clusters, as there are not enough vertices to make the basic shapes. The graphs which seem to perform best have a moderate variability in their cluster size, and many clusters with eight vertices or more.

The original goal of making the nodes completely "self-organising" as in autonomic networks has not yet been realised. The hard-coding of shape templates is a temporary solution until more sophisticated behaviour can be achieved.

Conclusions

We have presented work in progress on *graphemes*, a novel application of visual clustering applied to graph layouts. This system reduces complex graphs into sets of nodes arranged to form various simple geometric shapes, which is aimed to make it easy to remember the layout of a graph drawing, and also to make accurate estimates as to its overall size.

A large-scale study of the relative abilities of viewers to successfully estimate the number of nodes in the graph and recall the layout of clusters is ongoing. Early results suggest that users indeed can more easily recall the layout of a graph at a high level of abstraction by remembering the relative positions of the shapes.

Acknowledgements: This work was supported, in part, by Science Foundation Ireland under grant 03/CE2/I303_1 to Lero — the Irish Software Engineering Research Centre (www.lero.ie)

References

[1] Branke, J. (2001). *Dynamic graph drawing*. Springer Lecture Notes In Computer Science, 228–246.

[2] Buzan, T. & Buzan, B. (2000). *The mind map book*. BBC.

[3] Dijkstra, E. W. (1959). *A note on two problems in connexion with graphs*. Numerische Mathematik 1: 269–271.

[4] Dunbar, R. I. M. (1992). *Neocortex size as a constraint on group size in primates*. Journal of Human Evolution, 20, 469–493.

[5] Eades, P. (1984). *A heuristic for graph drawing*. Congressus Numerantium, 42(149160), 194–202.

[6] Eades, P. & Huang, M. L. (2000). *Navigating Clustered Graphs using Force-Directed Methods*. Journal of Graph Algorithms and Applications, **4**(3), 157–181.

[7] Eades, P., Lai, W., Misue, K., & Sugiyama, K. (1991). *Preserving the mental map of a diagram*. Proceedings of Compugraphics, **91**(9), 24–33.

[8] Girvan, M. and Newman, M. E. J. (2002). *Community structure in social and biological networks*. Proceedings of the National Academy of Sciences of the United States of America, **99**(12): 7821-7826.

[9] Herman, I., Melancon, G., & Marshall, M. S. (2000). *Graph Visualization and Navigation in Information Visualization: A Survey*. IEEE Transactions on Visualization and Computer Graphics, **6**(1), 24–43.

[10] Miller GA (March 1956). *The magical number seven plus or minus two: some limits on our capacity for processing information*. Psychological Review **63** (2): 81–97.

[11] Munzner, T. (2000). *Interactive visualization of large graphs and networks*. Ph.D. Dissertation.

[12] O'Madadhain, J., Fisher, D., Smyth, P., White, S., & Boey, Y. (2005). *Analysis and visualization of network data using JUNG*. Journal of Statistical Software.

[13] Sablowski, R. & Frick, A. (1997). *Automatic graph clustering*. LNCS, 395–400.

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

[1] Figure 2 courtesy of Jean-Louis Zimmermann, used with permission.