Using adjacency matrices to lay out larger small-world networks

GIBSON, Helen <http://orcid.org/0000-0002-5242-0950> and VICKERS, Paul

Applied Soft Computing 42 (2016) 80–92

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

Using adjacency matrices to lay out larger small-world networks

Helen Gibson[a,*], Paul Vickers[b,1]

[a] CENTRIC, Sheffield Hallam University, Cantor Building, 153 Arundel Street, Sheffield S1 2NU, UK
[b] Northumbria University, Department of Computer Science and Digital Technologies, Pandon Building, Camden Street, Newcastle-upon-Tyne NE2 1XE, UK

ARTICLE INFO

Article history:
Received 4 January 2015
Received in revised form
12 November 2015
Accepted 20 January 2016
Available online 29 January 2016

Keywords:
Small world networks
Adjacency matrix
Node attributes
Graph visualization
Targeted projection pursuit

ABSTRACT

Many networks exhibit small-world properties. The structure of a small-world network is characterized by short average path lengths and high clustering coefficients. Few graph layout methods capture this structure well which limits their effectiveness and the utility of the visualization itself. Here we present an extension to our novel graphTPP layout method for laying out small-world networks using only their topological properties rather than their node attributes. The Watts–Strogatz model is used to generate a variety of graphs with a small-world network structure. Community detection algorithms are used to generate six different clusterings of the data. These clusterings, the adjacency matrix and edgelist are loaded into graphTPP and, through user interaction combined with linear projections of the adjacency matrix, graphTPP is able to produce a layout which visually separates these clusters. These layouts are compared to the layouts of two force-based techniques. graphTPP is able to clearly separate each of the communities into a spatially distinct area and the edge relationships between the clusters show the strength of their relationship. As a secondary contribution, an edge-grouping algorithm for graphTPP is demonstrated as a means to reduce visual clutter in the layout and reinforce the display of the strength of the relationship between two communities.

1. Introduction

Small-world networks are a commonly occurring graph structure characterized by short average path lengths and high clustering coefficients [1]. This means that even when the network is large there are very few steps between each pair of nodes. Despite their prevalence very few methods are able to lay them out such that their structure is communicated optimally [2]. Examples of networks that display these characteristics in the real-world include social networks [3], biological networks [4] and even geophysical ones [5].

The high clustering coefficient of small-world networks provides an interesting problem for layout particularly as it has been shown that users seek to lay out graphs such that their clustered structure is apparent [6]. Thus, ensuring that the clustered structure of the graph is well represented in the layout seems crucial for enhancing users' understanding of the graph in the context of its clusters and the relationships between them. van Ham and van Wijk [7] have proposed one of the few small-world network specific layout methods while Gibson and Faith [8] put forward a method based on node-attribute data.

How clusters are represented in a graph is extremely important as the human perceptual system will naturally cause users to assume that there is a relationship between nodes that are placed close together [9]. For graphs that are highly clustered adhering to this principle when laying out the graph is key for communicating the structure of the network. Generally, force-directed layouts do not produce an accurate representation of small-world network structures. This is because they try to optimize the layout to have uniform edge lengths but longer edge lengths are usually required to separate clusters [2].

Force-directed methods such as LinLog (linear-logarithmic) [2,10] and OpenOrd (a successor to VxOrd) [11] do try to optimize for clustering based on topology while there are other methods that use attributes or pre-computed clusterings. Muelder and Ma's treemap [12] and space-filling [13] approaches use a pre-computed clustering, while the group-in-the-box layout [14] can take any user input or pre-computed clustering.

graphTPP (graph targeted projection pursuit) [8] is a method, encapsulated in an interactive software program, that has previously been used for the layout of small-world networks using node-attributes. The graph is laid out, assisted by direct user interaction, according to the specified clustering resulting in a clear visual separation of the clusters. Here we propose that rather

* Corresponding author. Tel.: +44 0114 225 6819; fax: +44 0114 225 6702.
E-mail addresses: h.gibson@shu.ac.uk (H. Gibson), paul.vickers@northumbria.ac.uk (P. Vickers).
[1] Tel.: +44 0191 243 7614..

http://dx.doi.org/10.1016/j.asoc.2016.01.036

than using node-attributes, the adjacency matrix of the graph can replace the multidimensional matrix of attributes and by significantly increasing the size of the graph (in terms of the number of nodes) demonstrates that graphTPP is scalable beyond the very small examples used in the previous work.

The paper proceeds as follows. Section 2 discusses related work on the layout of small-world networks. Section 3 introduces the Watts–Strogatz model that is used for computing the small-world networks and the community detection algorithms used by the layout. Section 4 presents the graphTPP layouts of the small-world networks for both one and two-dimensional Watts–Strogatz models whilst comparing them to results obtained using the OpenOrd and ForceAtlas layout algorithms. It also introduces an edge-grouping technique for reducing visual clutter caused by the edges in the graphTPP layout. Section 5 discusses the results and the limitations of this work while also recommending directions for future work. Section 6 then concludes the paper.

This research shows that graphTPP outperforms OpenOrd [11] and ForceAtlas [15] as a method for laying out small-world networks where the aim is to optimize the layout for the communities detected through various community detection algorithms. The main contribution then is the demonstration that graphTPP can be a viable layout method even when there is no typical node-attribute data available upon which to base the layout.

2. Related work

2.1. Small-world networks

Small-world networks are characterized by short average path lengths (the shortest path between any pair of nodes) and high clustering coefficients (e.g., in social networks this would be the number of friends a person has who are also friends, i.e., they complete the triangle). The average path length only grows logarithmically with the number of nodes, while there are many cliques or near cliques.

One of the most notable examples of a small-world network is from Milgram's [16] six degrees of separation experiment which proposed that most people in the United States at that time were separated by only six people in a chain of friendship. Recently, Boldi and Vigna [17] have modelled the degree of separation in the Facebook graph to be only 3.74. There are multiple other examples of small-world networks occurring in the real-world including social networks [3], biological networks [4] and geophysical networks [5]. Albert and Barabási [4] have hypothesized that the prevalence of networks in biology with small-world properties is due to their inherent structural advantages.

There are a number of ways to model a small-world network, the most popular being the Watts–Strogatz model [1]. The Watts–Strogatz model requires the construction of a regular ring lattice followed by random rewiring of the edges according to a rewiring probability p. This produces a graph with short average path lengths and a high clustering coefficient; however, compared to real-world small-world graphs the degree distribution does not tend to be scale-free (i.e., does not follow a power law distribution). The Barabási-Albert [4] model does produce a scale-free network but it does not exhibit the clustering properties that are integral to small-world networks and essential for this visualization technique.

Given that small-world networks are such a commonly occurring graph structure it is surprising that little attention has been paid to their visualization. Their structural properties are not well suited to general force-based layout methods since these methods often try to map shortest path lengths to Euclidean distances. The short average path lengths possessed by small-world networks result in high degree nodes being placed close to the centre of the graph and this encourages the well-known hairball-style layout to form (for an example in this paper see Fig. 9(c)). Further, many force-based techniques also try to optimize the layout according to certain aesthetic criteria. One such criterion is uniformity of edge length; however, long edge lengths are required to separate clusters and as such the force-style layouts do not well represent the clusters formed by the small-world network [2].

Ten years have passed since Auber et al. [18] commented that the "the structural properties of small-world networks have not yet been fully exploited from a visualization perspective" but to our knowledge still very few techniques exist that deal specifically with layout of graphs exhibiting small-world properties. Auber et al.'s [18] multi-scale small-world layout is one solution. It requires decomposing the graph in a hierarchical manner by rating the strength of each edge and removing the so-called 'weaker' ones. This helps with exploring the individual clusters.

Other small-world network layouts include van Ham and van Wijk's [7] that uses an adapted version of Noack's [2] force-model to further separate the clusters while van Ham and Wattenberg [19] create a sparse backbone of the graph by removing edges with low betweenness centrality, laying out the graph at this level and then proceeding to add edges back in. Topolayout [20] also detects small-world features while HiMap [21] has adapted Kamada and Kawai's [22] algorithm to produce clustered layouts.

Recently, Gibson and Faith [8] used a projection based technique to lay out small-world networks based on node-attributes aiming to produce a clustered layout. Here we extend that technique to a much larger class of small-world networks and no longer rely on the node-attributes but instead use the topological structure of the graph itself.

2.2. Graph layout

There are many hundreds of solutions to the graph layout problem each proposing a different method to highlight different features of the graph. While force-based techniques are extremely popular, they are generally not suitable for small-world networks for the reasons detailed above. There are, however, a few force-based methods that try to optimize for clustering and, hence, are potential candidates for the layout of small-world networks.

Noack's [2,10] LinLog layouts are one such example and they have been applied to the layout of small-world networks. The LinLog technique ignores the uniform edge length criterion in order to separate clusters. The name derives from the fact that the model has linear attractive forces but logarithmic repulsive forces. Noack has already shown that the method can better separate a graph into clusters compared to the Fruchterman–Reingold method [23]. OpenOrd [11] is another force-based method that aims to encourage clustering based on the simulated annealing algorithm. It is a multi-level method that cuts long edges to reduce the domination of the repulsive forces which improves clustering. It is a highly scalable algorithm that can lay out graphs with hundreds of thousands of nodes.

Similar to some force-based methods are those that use dimension reduction. Both classical and distance multidimensional scaling (MDS) use the dissimilarity matrix of the shortest paths between node pairs to generate the embedding while PivotMDS [24] is a sampling approximation technique to classical scaling where nodes are positioned according to the positions of a subset of pivot nodes. High-dimensional embedding (HDE) [25] is similar to PivotMDS but the final step uses principal component analysis to project the layout onto 2D space.

Other methods that make use of dimension reduction are EdgeMaps [26] which relies on node-attributes and an MDS

projection for the layout and PEx-graph [27] which can either use attributes or the graph's connectivity for layout.

Because small-world networks are inherently clustered, cluster-based layout techniques are also appropriate. Methods such as Group-in-a-box [14] require a clustering to be pre-defined but they then place each cluster into its own bounded box and the nodes in each box (cluster) can have their own layout applied. Muelder and Ma also proposed treemap [12] and space-filling approaches [13] which use community detection algorithms to pre-compute a clustering first and then use the communities to determine the decomposition into the treemap or the order of placement for the space-filling method.

2.3. Targeted projection pursuit and graphTPP

Targeted projection pursuit (TPP) [28,29] is an open-source exploratory, visual, interactive dimension reduction technique incorporated into a piece of software, known as the TPP tool, that provides a GUI front-end that presents both a real-time visualization of the current projection and the ability for the user to interact with this projection directly.[2] The tool itself is built on top of the data mining software Weka [30] and through the TPP tool the user is able to explore the space of possible linear projections from a high-dimensional space onto two dimensions. The technique focuses on three areas: finding a projection that groups the points into specified clusters, identifying the discriminatory dimensions that can be used to describe and analyze the clusters and, thirdly, identifying outliers.

Through the interactive user interface, the user can separate nodes into clusters or drag them about in the two-dimensional space to fit their intuition or understanding of the data. The underlying TPP algorithm will then search for the projection that best matches the target view that the user wants to see.

Formally, a set of n entities is described by the $n \times k$ matrix X that defines each entity's position in k-dimensional space. A $k \times 2$ projection matrix P maps the entities onto two dimensional space. When the user defines an $n \times 2$ target view T, TPP searches for a projection that minimizes difference between this target view and the projection. That is

$$\min \|T - XP\|. \quad (1)$$

The projection matrix is found by training a single-layer perceptron artificial neural network with k inputs and two outputs. The n rows of the original matrix are examined in order and standard back-propagation is used to train the network to generate the rows of the target matrix T according to a least-squares calculation. Once convergence is reached the original data is transformed into the 2D view where the connection weight between each input neuron and the two output neurons gives the weight of each dimension in the final projection and thus the projection matrix.

graphTPP leverages the TPP algorithm for graph layout and extends the original TPP tool software to import graph data and updates the visual display and control panel to facilitate viewing and interaction with the graph. The original motivation for graphTPP was to use the attributes of the nodes for the dimension reduction process that creates the layout. Each node-attribute would describe a dimension of the data and given a clustering, either pre-defined or through the k-means algorithm, the aim was to lay out a graph in a clustered fashion such that the layout would be a direct result of the attributes, bringing the graph's topological structure and its attributes together. This would facilitate the understanding of the graph's structure from the point of view of the attributes.

Clustering, in particular, has been identified as an important structural feature to be represented in layout with users favouring it over traditional aesthetic criteria [6] while the use of attributes in layout can be used to create a deeper understanding of the graph [31]. graphTPP expressly aims to separate the graph into clusters.

The use of graphTPP here is based on the same principle of using a high-dimensional data matrix that describes some features of the graph's nodes but the attribute matrix is replaced by the adjacency matrix (see start of Section 4). The adjacency matrix associates each node with a column and a row. An entry is made in position (i, j) if node i is connected to node j. For example, in Fig. 1 the highlighted nodes and edge shown in the graph are circled in the table, i.e., the adjacency matrix. This layout method has more in common with the force techniques as both rely on the topological structure for layout. graphTPP is then able to leverage all the interactive features of the original TPP tool such as the automatic separation of points into clusters, direct user interaction as well as a number of options for adjusting the visual features displayed on the layout (such as node size, colour, shape etc.). graphTPP also incorporates new options for controlling the visual features including edge shape and appearance, node labelling and some filtering options not utilized by the graphs and layouts in this paper. Further explanation of the process of using graphTPP in this paper is given in Section 4.1.

3. Small-world networks

3.1. Network generation

The small-world networks are generated according to the Watts–Strogatz model as implemented in R package igraph [32]. The Watts–Strogatz model aims to generate a graph with a high clustering coefficient and a short average path length, thus simulating the characteristics of a small-world network.

The basic Watts–Strogatz small-world model assumes a ring lattice of n nodes with k connections per node. Each edge is then randomly rewired with a probability p where $p = 0$ would describe a regular network and $p = 1$ describes a completely random network. The graph has the structural properties $L(p)$ which describes path length and $C(p)$ which describes the clustering coefficient. When $p = 0$ the value of L grows linearly with n and is well clustered (high $L(p)$ and high $C(p)$) to form a large-world network. When $p = 1$ the random network is poorly clustered (low $L(p)$ and low $C(p)$) and L grows logarithmically with n. However, for intermediate values of p, $L(p)$ is almost as small as L_{random} while $C(p) \gg C_{random}$. Hence, it approximates the small-world properties of a short average path length and a high clustering coefficient.

The igraph package in R [33] includes an implementation of the Watts–Strogatz model which was executed by varying the following parameters

```
watts.strogatz.game(dim, size, neighbours, p)
```

where `dim` = dimension of the lattice (1 for a ring lattice); `size` = the number of nodes in each dimension; `neighbours` = num. nearest neighbour connections; `p` = the rewiring probability.

Table 1 shows the how the parameters were varied. A constant rewiring probability of 0.05 was used while only one and two-dimensional models were considered.

3.2. Community detection

In network science, community detection is the partitioning of a graph into clusters or communities. Generally the partitions are divided such that a node is more likely to be connected to other

[2] https://code.google.com/p/targeted-projection-pursuit/.

Table 1
The parameters used to generate the small-world networks and the number of communities detected by the community detection algorithms.

	Parameters					Community detection algorithms					
Nodes	Edges	Dim	Size	Nearest neighbours	Rewiring probability	Edge betweenness	Fast greedy	Leading eigenvector	Spinglass	Walktrap	Label propagation
500	5000	1	500	10	0.05	12	3	12	9	9	21
1000	5000	1	1000	5	0.05	18	6	27	10	20	74
1500	7500	1	1500	5	0.05	22	9	34	10	31	102
2000	4000	1	2000	2	0.05	51	46	45	10	93	243
2000	10,000	1	2000	5	0.05	30	7	47	10	38	150
3000	15,000	1	3000	5	0.05	–	10	–	10	–	–
4000	20,000	1	4000	5	0.05	–	13	–	10	–	–
5000	25,000	1	5000	5	0.05	–	14	–	10	–	–
400	4800	2	20	3	0.05	7	4	8	9	7	1
400	2400	2	20	2	0.05	8	4	7	8	9	10
625	1250	2	25	1	0.05	18	12	14	10	12	56
625	3750	2	25	2	0.05	11	4	10	10	10	25
900	5400	2	30	2	0.05	12	4	11	10	12	22
1225	7350	2	35	2	0.05	12	4	14	10	14	34
1600	9600	2	40	2	0.05	15	4	12	10	16	46

nodes in the same community than they are to those outside their community. Community detection is usually only based on the topological properties of the graph rather than on attributes. Some algorithms impose a limit on the number of communities the graph is partitioned into while others place no restrictions on the number.

At the most basic level, identifying communities in the graph helps in understanding the graph's structure which can enable the classification of a node's structural position, if there are nodes that are on the periphery of a cluster or potentially overlapping clusters. The communities may also communicate the hierarchical organization of the graph [34]. In this paper we use six different community detection algorithms included in the igraph package. These are edge-betweenness [35] (hierarchical decomposition based on iteratively removing edges with the lowest edge-betweenness centrality), fast-greedy [36] (merges nodes into communities in a greedy manner based on optimizing the modularity function), leading eigenvector [37] (iteratively divides the graph into communities based on the signs of the eigenvector that corresponds to the largest eigenvalue of the modularity matrix), Walktrap [38] (merges communities based on short random walks since staying within a cluster is more likely than leaving it), Spinglass [39] (statistical physics approach where a node takes one of the n spin states and states changes depending on the state of neighbouring nodes) and label propagation [40] (each node assigned one of the k labels and nodes take the most common label of their neighbours). Further explanations of the community detection algorithms are given in the supplemental material and a detailed review can be found in Fortunato [34].

4. Layout and comparison

Gibson and Faith [8] demonstrated graphTPP's capabilities for the layout of small-world networks. In that case the network was small with only 30 nodes clustered into four groups. However, it indicated the potential of graphTPP as a layout method for small-world networks. Here we extend that work by applying it to the layout of much larger networks.

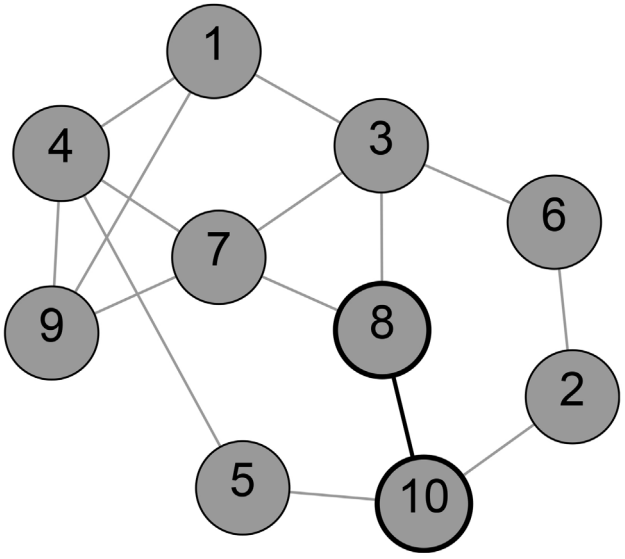
More importantly, since artificially generated networks do not have attributes the graph's adjacency matrix is used instead. This means that, like force-based methods, the layout now only relies on the topological structure of the graph. It is compared to two force-based techniques as implemented in the graph layout software Gephi: Martin et al.'s [11] OpenOrd layout method and the ForceAtlas [15].

The ForceAtlas layout is included as a comparison to a general non-optimized for clustering force-based layout whose performance should be better than the Fruchterman–Reingold algorithm but worse than LinLog in terms of clustering. Because of system unavailability it was not possible to compare performance against van Ham and van Wijk's framework [7].

The networks were generated using the Watts–Strogatz [1] model in the igraph package in R. Various clusterings are computed through community detection algorithms and recorded (see Table 1). Each graph created was the result of running one particular instance of Watts–Strogatz method. Due to the random rewiring probability each run of this method, even with the same parameters, will produce a slightly different graph, in terms of which nodes are connected to each other; however, this difference does not affect the validity of these results as each graph still has a small-world structure. The community detection algorithms were executed with their default parameters except for the Spinglass method which was run with a maximum of ten spin states which limited the number of communities to ten. This was done to ensure that there was at least one algorithm that was generating a cluster set that was not excessively large and a graphTPP layout could be produced. Out of the six community detection algorithms the Edge-Betweenness, Fast-Greedy, Leading Eigenvector and Walktrap algorithms consistently produce the same community when run over the same graph, in terms of number and size of communities created, while the Spinglass and Label Propagation algorithms produce slightly different results each time. However, these differences are not large and it was considered that testing using one particular example for these two algorithms was still sufficient for demonstration purposes.

4.1. Data preparation and import

Once the graphs had been created in R and each node had been assigned a community using each of the community detection algorithms the command `get.adjacency` was used to create the adjacency matrix of each graph and six additional columns were added identifying the community each node belonged to for each community detection algorithm. This was then exported to CSV ready to be converted into the ARFF file format supported by Weka. Two further files, a simple list of nodes and the communities they belong to and an edge list was also exported. An ARFF file is created by loading the CSV file into Weka and ensuring that the first n columns, where n is the number of nodes, are described as numeric columns (as these define the dimensions to be used in the projection). The next six columns are class-type columns which define the membership of nodes to particular communities for each community detection algorithm and the final column is a string column



(a)

Attributes

Nodes	1	2	3	4	5	6	7	8	9	10
1	0	0	1	1	0	0	0	0	1	0
2	0	0	0	0	0	1	0	0	0	1
3	1	0	0	0	0	1	1	1	0	0
4	1	0	0	0	1	0	1	0	1	0
5	0	0	0	1	0	0	0	0	0	1
6	0	1	1	0	0	0	0	0	0	0
7	0	0	1	1	0	0	0	1	1	0
8	0	0	1	0	0	0	1	0	0	(1)
9	1	0	0	1	0	0	1	0	0	0
10	0	1	0	0	1	0	0	(1)	0	0

(b)

Fig. 1. The graph shown in (a) has its adjacency matrix in (b). A connection between two nodes is indicated by a '1' at the intersection of the corresponding row and column. Two nodes 8 and 10 are linked and highlighted in bold in (a), the corresponding values in the adjacency matrix are then circled in (b).

representing the node's ID. This file is then exported from Weka as an ARFF file and can be imported into graphTPP. The edge list file can then also be imported so that graphTPP is able to represent the edges in the graph.

Once the ARFF and edge list file are imported into graphTPP the user can select a particular set of communities to try to separate the nodes. In the following cases once a community detection algorithm had been selected the 'Separate points' button was pressed and held down to begin an automatic separation of the communities. When such an action is chosen the graphTPP algorithm identifies a number of points in space (based on the number of communities to separate) where these points are maximally separated from one another. This becomes the target projection and the automatic layout progresses by trying to achieve this maximal separation. If a user believes that the automatic separation is not achieving the desired level of separation they can select a set of nodes, either individually, by community or using a rectangle selection. Once selected, the user can attempt to move this node around in order to achieve something closer to their desired target view. Fig. 2 shows how the graph and the layout appear in graphTPP. The figures presented in this paper are based on the SVG file exported directly from graphTPP. The code and data files needed to recreate these and the following steps can be found at https://github.com/helengibson/graphTPP.

The Force-Atlas layout and the OpenOrd layouts were produced by loading the nodes file (with the community attributes) and the edge file into the graph layout software Gephi and running each method with their default parameters and exporting to SVG using Gephi's export tool.

The next section explores the layouts produced using these three layouts for a number of different sized graphs and community detection algorithms.

4.2. One-dimensional models

The first network produced from the one-dimensional model contained 500 nodes and 5000 edges. The 5000 edges were produced by requiring each node to connect to its ten nearest neighbours and the small-world graph was produced by using a rewiring probability of 0.05.

Each community detection algorithm was run on the graph and a set of clusters was produced for each algorithm. The six algorithms produced cluster sets of varying sizes ranging from 3 to 21. The Walktrap community detection algorithm [38] is based on random walks and partitioned the graph into 9 clusters. Fig. 3 shows the graph with this Walktrap community detection algorithm applied for each of the three layout methods. Table 1 in the supplemental material shows the layout for all of the community detection algorithms for this graph.

Each layout is clearly influenced by the original ring structure from which the network was generated. However, only graphTPP is able to separate the clusters correctly and into their own distinct area. ForceAtlas also separates the clusters correctly but the snake-like layout means that they lead on to one another. This makes analysis of how each cluster is connected to the others difficult. Colour is also required to determine where each cluster starts and ends. The OpenOrd method also shows the clusters leading on to one another but the nodes are now positioned into tight groups. The algorithm actually subdivides them into smaller clusters than those detected by the Walktrap algorithm. Again, it is not clear without the colouring to which cluster each of these sub-clusters belongs. However, seeing these clusters could also stimulate further investigation into why they are sub-divided up by this method and, in particular, the significance of where the clusters overlap.

graphTPP is not only a tool for layout, it also facilitates analysis of the graph according to its attribute composition. Fig. 4 shows the contribution that four different nodes make to the layout. In Fig. 4(a) and (b) the red nodes are those possessing that particular attribute (in this case that means nodes which are connected to that particular node). Those highlighted attributes are clearly two of the main contributors to that specific cluster, i.e., that node connects to many nodes within that cluster. The attributes highlighted in Fig. 4(c) and (d) show these nodes connect equally to adjacent clusters. This indicates that those two nodes may be bridges between the two communities.

While a graph with 500 nodes is a significant increase on the graph used in Gibson and Faith [8] it is still fairly small. A further six-fold increase in the number of nodes has a marked

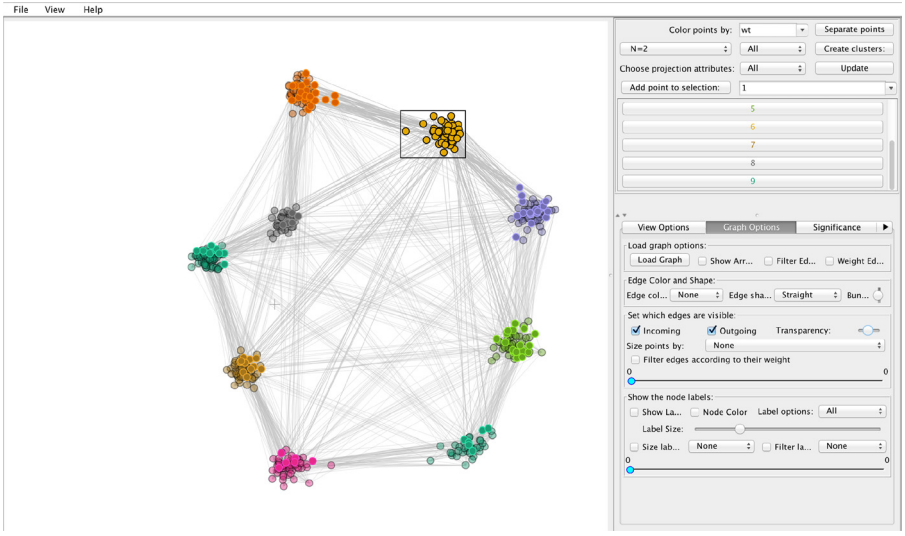
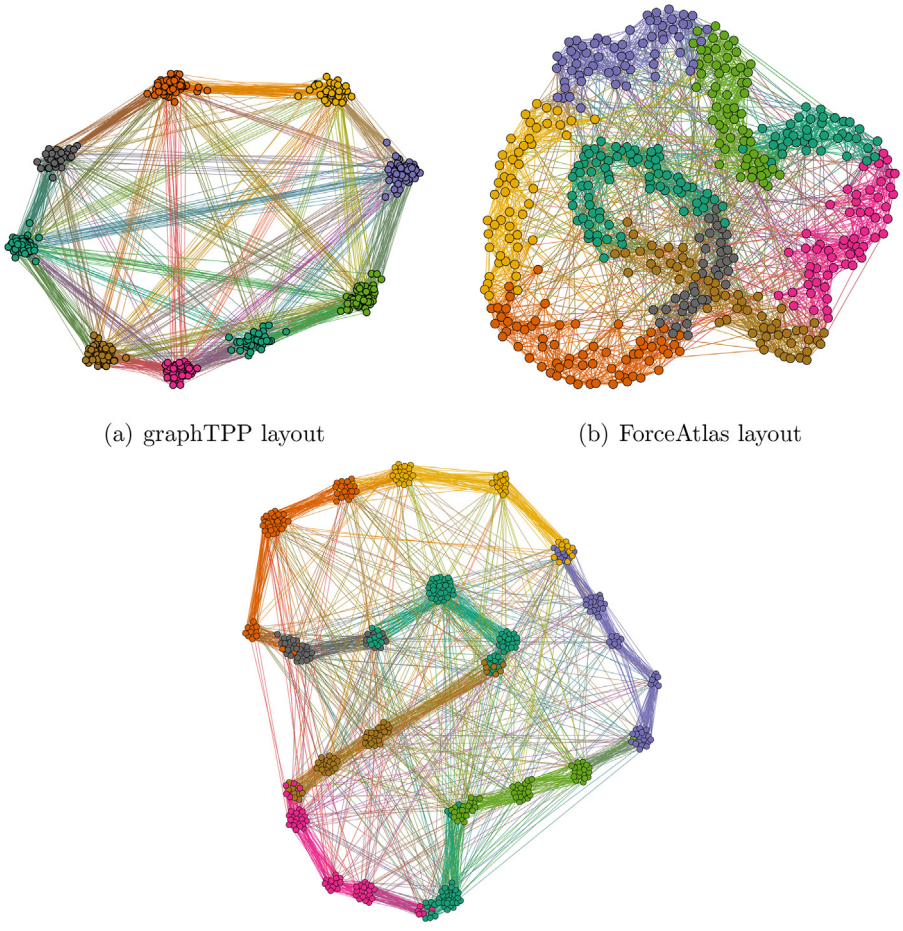


Fig. 2. A screenshot of the graphTPP layout interface where the number of nodes is 500 and the number of edges is 500. The right hand panel shows some of the options available to users when laying out the graph.

impact on the ability to separate nodes into clearly defined clusters for both the ForceAtlas and OpenOrd algorithms as shown in Fig. 5. With 3000 nodes and 15,000 edges graphTPP is still able to separate the clusters clearly. In this case there are 10 clusters which were again determined using the Walktrap algorithm. The clear separation enables some analysis of the strength of the relationships between different communities, further evidenced in Fig. 7.

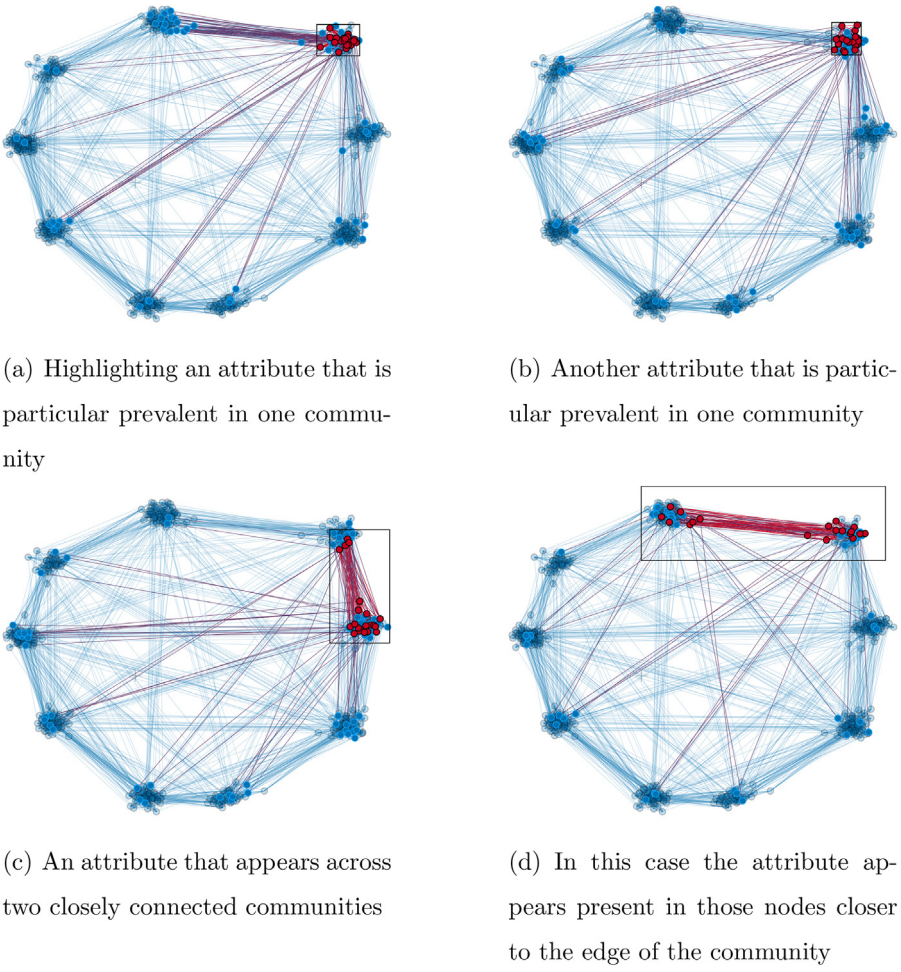


(a) graphTPP layout

(b) ForceAtlas layout

(c) OpenOrd layout

Fig. 3. The small-world graph with 500 nodes and 5000 edges laid out using the three different algorithms using the Walktrap community detection algorithm. The original ring lattice structure is clearly visible in all three layouts.



(a) Highlighting an attribute that is particular prevalent in one community

(b) Another attribute that is particular prevalent in one community

(c) An attribute that appears across two closely connected communities

(d) In this case the attribute appears present in those nodes closer to the edge of the community

Fig. 4. The same small-world graph with 500 nodes and 5000 edges as in Fig. 3(a). Four different attributes are selected and the nodes that have those attributes are highlighted. A red node indicates that that node has that particular attribute. Images (a) and (b) show how for those two example attributes, other nodes with the same attributes fall in the same cluster while images (c) and (d) show how nodes in adjacent clusters sometimes share attributes. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

However, for this graph it is more difficult to identify nodes which contribute the most to the determination of cluster membership. This is because the average degree of each node is 10 whereas each cluster contains, on average, 300 nodes. This does not affect the clustering ability of the graphTPP algorithm though.

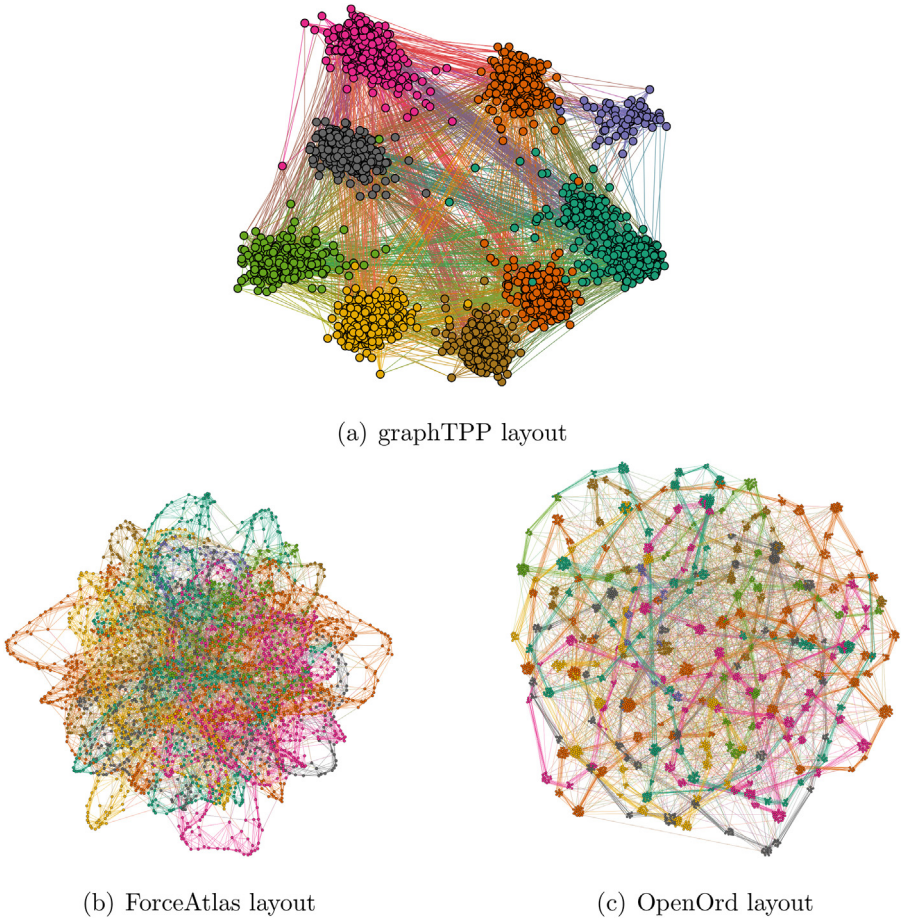
For this graph (Fig. 5), the volume of edges in the graphTPP layout overwhelms the overall layout of the graph and its clustered structure. This means that while the nodes are well clustered, the sheer number of edges impedes the ability to interpret the strength of the relationships between the clusters. Section 4.3 discusses a solution to this problem as an extension of the graphTPP tool.

Beyond 3000–4000 nodes it becomes difficult both to generate the graphs and calculate the communities. The large number of attributes that including every node as an attribute brings led to slow system performance; running graphTPP on a typical desktop computer meant that it was impractical to attempt to lay out graphs of 5000 nodes and above. Of course, increased processing power would mitigate this effect. Future work should also focus on carrying out a detailed performance analysis of the graphTPP algorithm to look for opportunities for code optimization. For this type of graph structure ~3000 nodes is the current practical limit for graphTPP operating in a regular desktop environment. Further layouts for graphs with 1000, 1500 and 2000 nodes can be found in the supplementary material.

4.3. An edge-grouping method

When experimenting with this layout method it quickly became clear that the volume of edges to be displayed was impacting on graphTPP's efficacy. For example, in Fig. 5(a) the clusters are well separated but the number of edges makes it difficult to interpret how strongly each cluster is connected. Thus, while the layout was successful in positioning the nodes, dealing with the edges was becoming a problem and distracting from the interesting structures on display.

Edge-bundling algorithms group edges to reduce visual clutter. Multiple edge-bundling algorithms have been proposed previously [41–43] but here we use an edge-grouping method that makes use of the intrinsic properties of the graphTPP layout, namely the clusters. The centroid of each cluster is found and an edge is drawn from each node to the centroid of its own cluster with its exact position offset slightly from the centroid in order to allow the thickness of the edge-group to be proportional to the number of edges it contains. When the bundle reaches the centroid of the other cluster the edges split from the edge-group and complete the edge, as shown in Fig. 6. Unlike in Fig. 6, in the actual graphTPP layout there are usually nodes placed over the centroid position and so the artefact of where the edges meet the edge-group is hidden. The bundled view is intended to improve the overview of the graph rather than for when individual edges are being explored.



(a) graphTPP layout

(b) ForceAtlas layout

(c) OpenOrd layout

Fig. 5. The small-world graph with 3000 nodes and 15,000 edges laid out using each of the three layout algorithms based on the fast greedy community detection algorithm with nine clusters. Here we clearly see that graphTPP is the only layout algorithm that is able to separate the clusters successfully while both the ForceAtlas and the OpenOrd layouts produce a much more tangled layout.

Fig. 7 shows the edge-grouped layout that corresponds to the layout in Fig. 5(a). The visual clutter is significantly reduced and it is much easier to see relationship strengths between the various clusters.

Using a slightly smaller example, with 2000 nodes and 10,000 edges and also with fewer clusters (7 compared to 10), the clarity the edge-groups offer becomes more obvious. Fig. 8 shows the graphTPP layout for this graph with the original edges and with the grouped edges side-by-side. In Fig. 8(a) it is very difficult to distinguish that the volume of edges between the clusters differs depending on which pair of clusters is chosen whereas in Fig. 8(b) it is much clearer. For example, the edge-groups linked to cluster 2 in Fig. 8(b) appear thicker than those belonging to most of the other clusters. Table 2 shows the total number of edges between each cluster in this 2000 node graph which supports this conclusion since the values show that cluster 2 has either the highest or second highest number of connections to each of the other six clusters. The edge-grouped version of graphTPP was able to show this cluster's high connectivity

Fig. 6. Example of the grouping method. Edges are drawn from each node to a position slightly offset from the cluster's centroid. From here the edges are grouped and drawn to the centroid of the corresponding cluster where the group then splits apart again to make the individual connections.

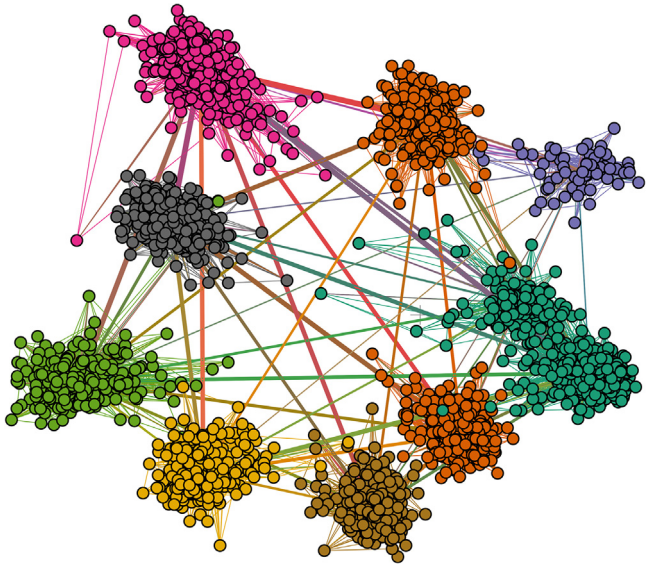


Fig. 7. The corresponding graphTPP edge-grouped layout for the small-world graph with 3000 nodes and 15,000 edges shown in Fig. 5(a). The edge-groups reduce the clutter on display and make the graph easier to interpret while keeping the clustered structure.

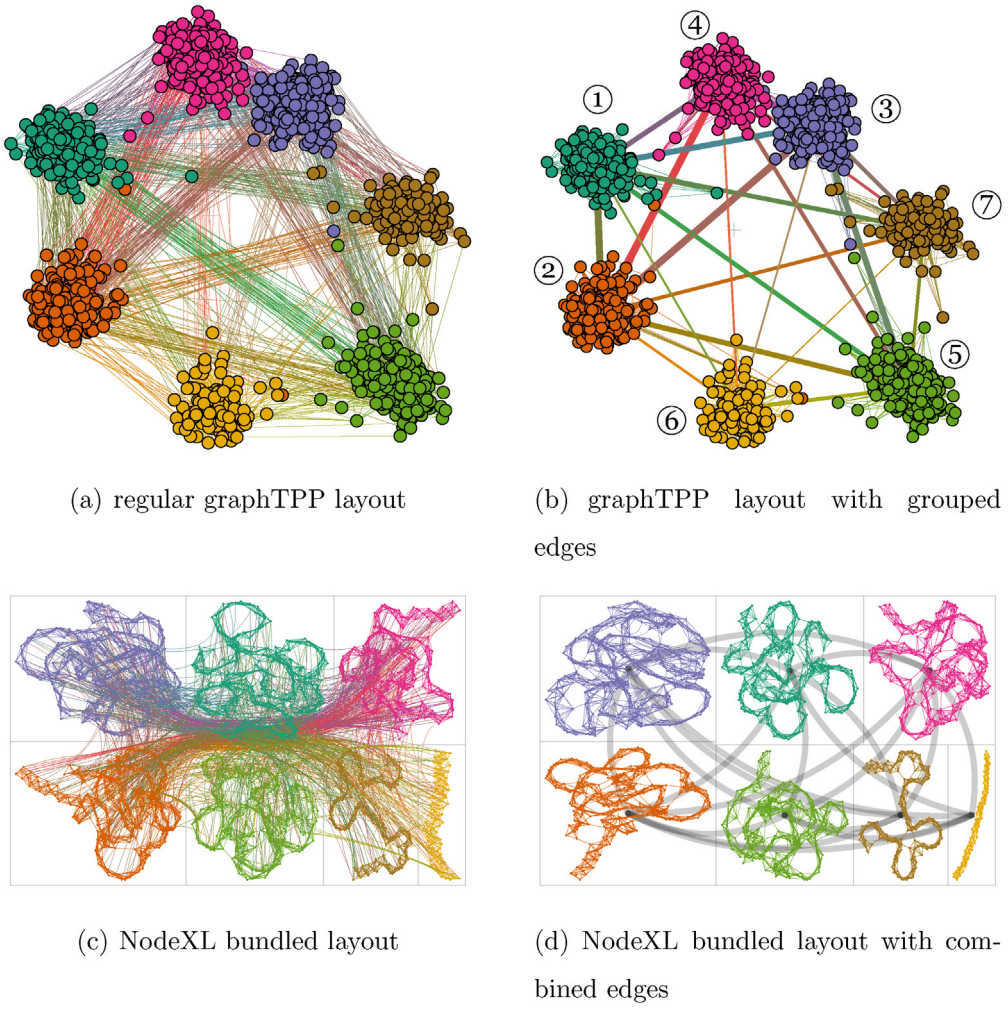


Fig. 8. The one-dimensional model layout with 2000 nodes and 10,000 edges. The nodes are grouped into seven communities by the fast-greedy community detection algorithm. (a) The regular graphTPP layout while (b) the layout with grouped edges. (c) and (d) The graph grouped by the same clusters with each cluster individually laid out using Harel and Koren's [44] fast multiscale method in NodeXL. (c) Uses bundled edges while (d) uses combined edges.

immediately while the original graphTPP version could only cluster the nodes but could not expressly distinguish how well connected they were.

Fig. 8(c) and (d) also shows this 2000 node graph laid out in NodeXL [45] using the Group-in-a-box layout [14] for the main layout and Harel and Koren's fast multi-scale layout [44] for the within box layouts. This is a method that rigidly adheres to the clustered structure by constraining each cluster to its own box. Fig. 8(c) uses an edge bundling algorithm to display the edges while Fig. 8(d) uses a combined edge method which does not take into account the number of edges between each cluster. In terms of understanding the relationships between the clusters neither layout is able to do this as well as graphTPP although the layout in Fig. 8(d) is able to show some of the within-cluster structure.

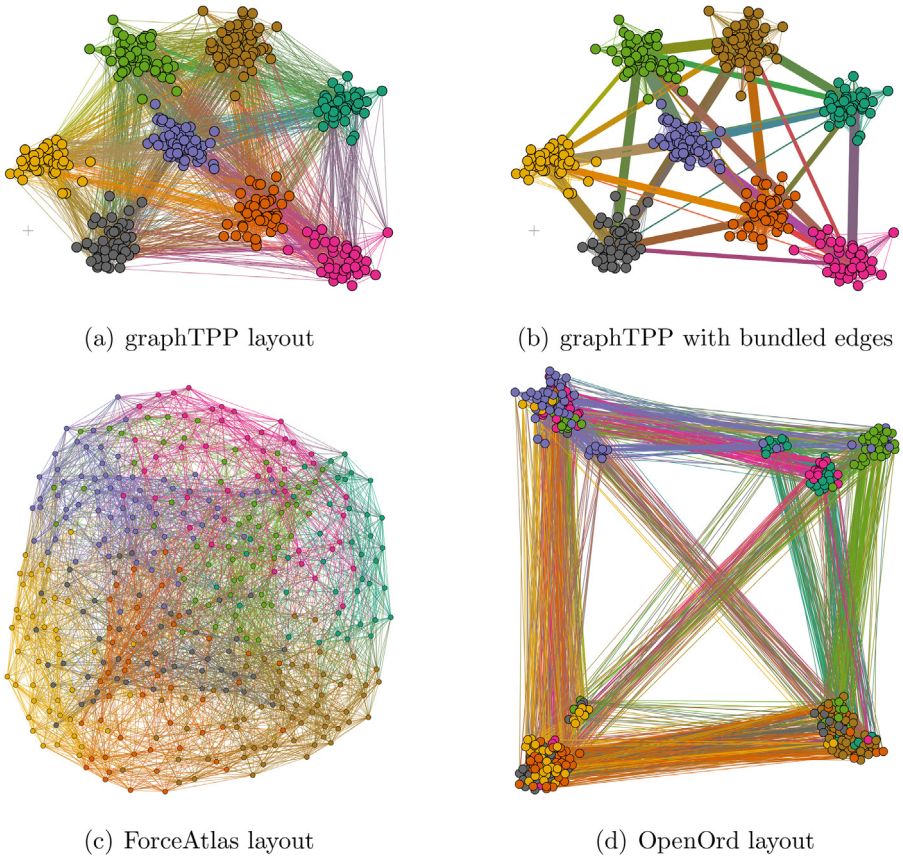
4.4. Two-dimensional models

The two-dimensional model incrementally increases the number of nodes along each dimension. For example, a size of 20 gives 400 nodes (20×20). Here we test models up to a maximum of 1600 nodes (40×40). In this model, setting the nearest neighbours parameter to two balanced the requirement of having enough edges so that they could be used as attributes without overloading the graph.

Table 2
Number of edges between each cluster for the 7 clusters defined by the fast-greedy algorithm for the graph with 2000 nodes and 10,000 edges. The final column shows the total number of nodes in each cluster.

Cl.	1	2	3	4	5	6	7	Size
1	1520	72	59	50	44	19	46	334
2	72	1758	71	75	54	24	36	384
3	59	71	1782	75	61	16	39	389
4	50	75	75	1347	39	20	27	298
5	44	54	61	39	1337	38	33	293
6	19	24	16	20	38	441	9	102
7	46	36	36	27	33	9	908	200

The initial 400 node model had 4800 edges and the community detection algorithms divided the graph into sets of clusters of reasonable sizes for each of the algorithm; the algorithms yielded 7, 4, 8, 9, 7 and 1 communities respectively (see corresponding row in Table 1). Using the leading eigenvector community detection algorithm [37], Fig. 9 shows the ForceAtlas layout struggling to produce anything other than what could be described as a hairball layout and although the OpenOrd algorithm shows a clear structure, it is one that does not seem to correspond to any of the community detection algorithms used in this case (see Table 6 in the supplemental material for further comparisons). graphTPP was again able to cluster the nodes into their communities and the bundling of



(a) graphTPP layout (b) graphTPP with bundled edges

(c) ForceAtlas layout (d) OpenOrd layout

Fig. 9. The small-world graph with 400 nodes and 4800 edges laid out using each of the layout three algorithms based on the leading eigenvector community detection algorithm which detects eight clusters. graphTPP is clearly able to cluster the graph into the communities and the grouped edges in (b) provides useful additional information that demonstrates how strongly each of the clusters are connected.

edges brings out this structure even more clearly, preventing the edges from overpowering the graph.

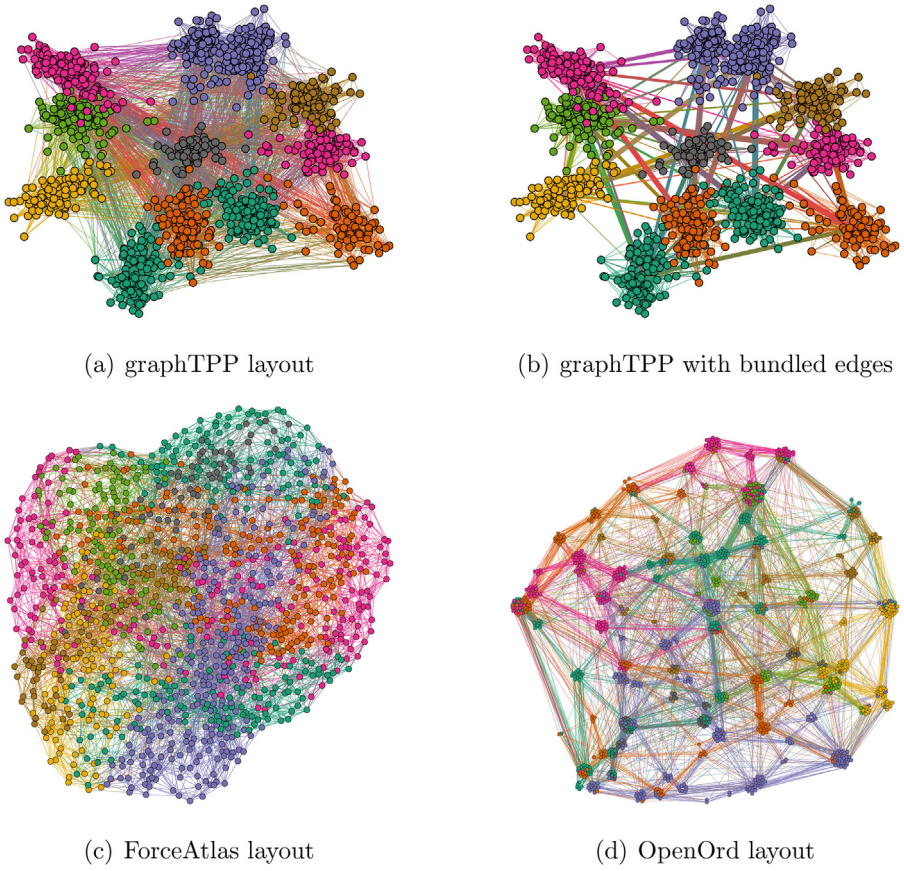
The graphs produced from the two-dimensional model have a lower limit on the maximum number of nodes that graphTPP can lay out compared to the one-dimensional model. In this case 1225 nodes with 7350 edges was the maximum. As can be seen in Fig. 10, which displays the clustering found by the edge-betweeness algorithm, ForceAtlas was again unable to produce a layout that clearly represented the structure of the graph. The OpenOrd algorithm is able to show some structure although it divides the nodes into many different communities. graphTPP displays a layout which clearly respects the communities detected by the edge-betweenness algorithm. This is particularly useful if the aim is to understand the relationships between the different communities because the thicker the bundle the greater the number of edges between any two communities. Further examples of layouts are shown in the supplemental material for graphs with 400, 625 and 900 nodes.

In this section we have shown how graphTPP is able to layout small-world graphs which respect a community clustered structure in such a way that the community structure is communicated through the layout of the graph. We have also demonstrated, in terms of the visual separation between each cluster, that graphTPP is able to produce this type of layout more reliably and consistently than two other layout methods: Force Atlas and OpenOrd. Thus, laying out a small-world network with graphTPP allows us to (1) view the clustered structure of the graph visually; and (2) from this clear visual separation, we can better understand the relationships between different clusters. One of the main aims of graph layout is to better understand the relationships between the different nodes in the graph and to do this visually. Therefore, since graphTPP provides this ability to do this, through an interactive platform, it has the potential to be applied to a number of different, real-world, small-world networks and generate insights about such graphs. Thus, although the Force-Atlas and OpenOrd were often able to produce layouts that appeared aesthetically pleasing these layouts were not able to communicate the clustered structure in the same way that graphTPP was. Furthermore, while graphTPP has its limits in terms of the size of graph that it can lay out, the degradation in layout quality for lager graphs was much more apparent for the Force-Atlas and OpenOrd graphs than it was for graphTPP despite the fact that overall Force-Atlas and OpenOrd are able to lay out graphs of a larger size.

5. Discussion

The structure of a small-world network should make it perfectly suited to visualization. Representing the communities that form within the graph should be a principal aim for layout if we want the visualization to communicate this structure effectively. The original graphTPP relied largely on node-attributes to drive the layout methods [8]. The highly clustered structure that characterizes small-world networks meant that there was potential to adapt this layout method by replacing the node-attribute matrix with the symmetric adjacency matrix of the graph. Since the edges, and hence entries in the attribute matrix, should be concentrated within clusters graphTPP was a good candidate for graph layout.

In this paper we have shown that for small-world networks generated by the Watts–Strogatz model in both one and two dimensions graphTPP was successful (in terms of visual



(a) graphTPP layout (b) graphTPP with bundled edges

(c) ForceAtlas layout (d) OpenOrd layout

Fig. 10. The small-world graph with 1225 nodes and 7350 edges laid out using each of the layout three algorithms based on the edge-betweeness detection algorithm.

presentation, visual distinction of clusters and representation of the clusters) in clustering the nodes into their appropriate predefined communities and, in doing so, producing a graph layout that outperformed other layout algorithms in terms of representing this structure.

While both the force-based methods initially clearly showed the structure of the communities in their layouts, as the graphs grew larger and more complex the quality of the techniques degraded. Even with the smallest of the two-dimensional models the ForceAtlas method was not able to produce a layout with any kind of structure while the OpenOrd method usually further divided the clusters or even divided them completely differently.

The edge-grouped graphTPP layout further enhanced the appearance and utility of the graphTPP layout by removing the visual clutter caused by the edges which had a tendency to overload the representation. The edge-groups provide a further advantage by showing how strongly each cluster is connected to the others and this can be used to determine the more important clusters in the graph and those which have more influence.

5.1. Limitations of small-world network layout with graphTPP

Despite the advantages of the graphTPP layout we have described in the last few sections, there are still a number of limitations associated with the graphTPP method of laying out graphs.

When importing data into graphTPP there is no limit on the number of clusters into which the nodes can be divided or how many it can attempt to spatially separate; however, a limitation, especially with this small-world structure, is that the clusters begin to become equally spaced over the display space. This results in graphTPP losing some of its power in being able to show the structure of the graph.

One of the goals of using TPP and the attribute-based graphTPP is to identify the most common attributes that occur in each cluster. In typical uses of TPP only a few attributes emerge as being significant and so these attributes can be used to define the cluster and form a useful part of the analysis. A limitation in this case, since we are now using the nodes as attributes and the edges as the attribute values, is that the set of significant attributes for each cluster is much larger. This is because each node only connects to a few other nodes and, therefore, each attribute only has a few non-zero values. In terms of analysis this means that when a cluster is analyzed its most significant attributes are most likely to be the nodes contained in that cluster thus telling us little new about the graph from an attribute analysis perspective.

The clustering of the nodes also causes a further issue. As can be seen in Table 2 the number of within cluster edges is far higher than between any of the clusters. This is a good thing in the sense that it is what makes the graphTPP layout algorithm work so well for this type of graph but this tight clustering comes at the expense of being able to see any of these within cluster edges and instead they are obscured by the nodes. However, this is not just a problem limited to graphTPP and can occur in any layout that clusters nodes closely together. The Group-in-a-box layout showed the opposite problem to this where the within cluster structure was clearer but the relationships between the clusters less so.

One of the difficulties of representing these small-world graphs and, in particular, their edges is that each community is usually connected to every other community. This means that even in the edge-grouped layout there are a lot of displayed edges which can

make the graph look cluttered and disorganized. In effect, this bundled layout is actually the layout of a small clique where each cluster would be represented by a single node and connected to every other node. This problem is exacerbated as the number of communities increases. The advantage that the graphTPP layout maintains over other layouts is that the proximity of the clusters is related to the topology and each one can be analyzed for each node's contribution. A potential solution would be to filter edges based on bundle size and only show edges which are part of a bundle over a certain size.

A limitation on the computation side is that due to the volume of attributes graphTPP is not able to keep up with the calculations and so although the layout is still interactive the interaction no longer happens in real-time and there is a gap between the user instructing the nodes to move and when they actually move. They also tend to 'jump' across the screen rather than move smoothly. This diminishes the interaction experience of graphTPP and the intermediate stages of interactive exploration.

5.2. Future research directions

As mentioned in the previous section, the ability to filter out some of the weaker bundled edges may also aid the clarity and usefulness of the layout. These edges would still be used in the projection but not displayed.

Further investigation into other methods beyond matrix diagrams [46] as a way to show the density of within cluster edges would aid not only this visualization method but also any others which group nodes into clusters but then suffer from the occlusion of within cluster edges. This would allow the user to investigate if there are any sub-patterns within the cluster or particular features that may have not been apparent when the nodes are tightly clustered together.

Determining if there is a specific set of tasks that are most easily accomplished with this type of layout would also provide an advantage. In particular, this kind of layout seems to support overview based tasks and in this case, it would mean that the user would know that they could lay out the graph using the graphTPP technique when they had a specific query or line of investigation in mind. This paper has already shown that assessing the strength of the connectivity between two clusters is one such potential task.

6. Conclusions

This paper has presented an extension to the small-worlds pilot study presented in Gibson and Faith [8] where graphTPP was used to lay out a small-world network using node attributes. In this case the number of nodes has been significantly increased but graphTPP still shows a superior ability to produce a layout that reflects the clustered structure of the graph compared to two force-based methods. The addition of the edge bundling method means that the strength of association between two communities is clearly visible. The use of the community detection algorithms resulted in reasonably equally sized clusters which also aided the analysis and visualization. It also showed that small-world networks are compatible with re-purposing edges as attribute values, thus graphTPP could, in theory, be applied as a viable layout option for any graph regardless of the existence of pre-existing attributes or not.

7. Supplemental material

7.1. Community detection algorithms

A more detailed description of each of the community detection algorithms can be found in the supplementary material.

7.2. All layouts

Further examples of layouts of graph of different sizes clustered using the community detection algorithms, as detailed in Table 1, can be found in the supplementary material.

Appendix A. Supplementary data

Supplementary data associated with this article can be found, in the online version, at http://dx.doi.org/10.1016/j.asoc.2016.01.036.

References

[1] D.J. Watts, S.H. Strogatz, Collective dynamics of 'small-world' networks, Nature 393 (1998) 440–442.
[2] A. Noack, An energy model for visual graph clustering, in: G. Liotta (Ed.), Graph Drawing, Lecture Notes in Computer Science, vol. 2912, Springer Berlin/Heidelberg, 2003, pp. 425–436.
[3] O. Sporns, D.R. Chialvo, M. Kaiser, C.C. Hilgetag, Organization, development and function of complex brain networks, Trends Cogn. Sci. 8 (9) (2004) 418–425.
[4] R. Albert, A.-L. Barabási, Statistical mechanics of complex networks, Rev. Mod. Phys. 74 (1) (2002) 47–97.
[5] X.-S. Yang, Small-world networks in geophysics, Geophys. Res. Lett. 28 (13) (2001) 2549–2552, http://dx.doi.org/10.1029/2000GL011898.
[6] F. van Ham, B. Rogowitz, Perceptual organization in user-generated graph layouts, IEEE Trans. Vis. Comput. Graph. 14 (6) (2008) 1333–1339.
[7] F. van Ham, J. van Wijk, Interactive visualization of small world graphs, in: IEEE Symposium on Information Visualization, 2004, pp. 199–206.
[8] H. Gibson, J. Faith, Node-attribute graph layout for small-world networks, in: 15th International Conference on Information Visualisation, 2011, pp. 482–487.
[9] C. McGrath, J. Blythe, D. Krackhardt, Seeing groups in graph layouts, Connections 19 (2) (1996) 22–29.
[10] A. Noack, Energy models for graph clustering, J. Graph Algorithms Appl. 11 (2) (2007) 453–480.
[11] S. Martin, W.M. Brown, R. Klavans, K.W. Boyack, OpenOrd: an open-source toolbox for large graph layout, in: Proc. SPIE 7868, 2011, pp. 786–806.
[12] C. Muelder, K.-L. Ma, A treemap based method for rapid layout of large graphs, in: 2008 IEEE Pacific Visualization Symposium, 2008, pp. 231–238.
[13] C. Muelder, K.L. Ma, Rapid graph layout using space filling curves, IEEE Trans. Vis. Comput. Graph. 14 (6) (2008) 1301–1308.
[14] E.M. Rodrigues, N. Milic-Frayling, M. Smith, B. Shneiderman, D. Hansen, Group-in-a-box layout for multi-faceted analysis of communities, in: Third IEEE Conference on Social Computing, 2011.
[15] M. Jacomy, S. Heymann, T. Venturini, M. Bastian, ForceAtlas2, A graph layout algorithm for handy network visualization (draft). http://www.medialab.sciencespo.fr/publications/Jacomy_Heymann_Venturini-Force_Atlas2.pdf.
[16] S. Milgram, The small world problem, Psychol. Today 2 (1) (1967) 60–67.
[17] P. Boldi, S. Vigna, Four degrees of separation, really, CoRR abs/1205.5509.
[18] D. Auber, Y. Chiricota, F. Jourdan, G. Melancon, Multiscale visualization of small world networks, in: IEEE Symposium on Information Visualization, 2003, p. 10.
[19] F. Van Ham, M. Wattenberg, Centrality based visualization of small world graphs, Comput. Graph. Forum 27 (3) (2008) 975–982.
[20] D. Archambault, T. Munzner, D. Auber, Topolayout: multilevel graph layout by topological features, IEEE Trans. Vis. Comput. Graph. 13 (2) (2007) 305–317.
[21] L. Shi, N. Cao, S. Liu, W. Qian, L. Tan, G. Wang, J. Sun, C.-Y. Lin, Himap: Adaptive visualization of large-scale online social networks, in: IEEE Pacific Visualization Symposium, 2009, PacificVis '09, 2009, pp. 41–48.
[22] T. Kamada, S. Kawai, An algorithm for drawing general undirected graphs, Inf. Process. Lett. 31 (1) (1989) 7–15.
[23] T.M. Fruchterman, E.M. Reingold, Graph drawing by force-directed placement, Softw.: Pract. Exp. 21 (11) (1991) 1129–1164.
[24] U. Brandes, C. Pich, Eigensolver methods for progressive multidimensional scaling of large data, in: M. Kaufmann, D. Wagner (Eds.), Graph Drawing, Lecture Notes in Computer Science, vol. 4372, Springer Berlin/Heidelberg, 2007, pp. 42–53.
[25] D. Harel, Y. Koren, Graph drawing by high-dimensional embedding, J. Graph Algorithms Appl. 8 (2) (2004) 207–219.
[26] M. Dörk, S. Carpendale, C. Williamson, Visualizing explicit and implicit relations of complex information spaces, Inf. Vis. 11 (January (1)) (2012) 5–21.
[27] R.M. Martins, G.F. Andery, H. Heberle, F.V. Paulovich, A. Andrade Lopes, H. Pedrini, R. Minghim, Multidimensional projections for visual analysis of social networks, J. Comput. Sci. Technol. 27 (4) (2012) 791–810.
[28] J. Faith, R. Mintram, M. Angelova, Targeted projection pursuit for visualizing gene expression data classifications, Bioinformatics 22 (21) (2006) 2667–2673.
[29] J. Faith, Targeted projection pursuit for interactive exploration of high-dimensional data sets, in: 11th International Conference on Information Visualization, 2007, pp. 286–292.
[30] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, I.H. Witten, The WEKA data mining software: an update, ACM SIGKDD Explor. Newslett. 11 (1) (2009) 10–18.

[31] J. Heer, A. Perer, Orion: A system for modeling, transformation and visualization of multidimensional heterogeneous networks, Visual Analytics Science and Technology (VAST), 2011.
[32] G. Csardi, T. Nepusz, The igraph software package for complex network research, InterJ. Complex Syst. (2006) 1695, URL http://igraph.sf.net.
[33] R Core Team, R: A Language and Environment for Statistical Computing, R Foundation for Statistical Computing, Vienna, Austria, 2013, URL http://www.R-project.org.
[34] S. Fortunato, Community detection in graphs, Phys. Rep. 486 (3–5) (2010) 75–174.
[35] M.E.J. Newman, M. Girvan, Finding and evaluating community structure in networks, Phys. Rev. E 69 (2004) 026113.
[36] A. Clauset, M.E.J. Newman, C. Moore, Finding community structure in very large networks, Phys. Rev. E 70 (2004) 066111.
[37] M.E.J. Newman, Finding community structure in networks using the eigenvectors of matrices, Phys. Rev. E 74 (2006) 036104.
[38] p. Yolum, T. Güngör, F. Gürgen, C. Öztura (Eds.), Computer and Information Sciences – ISCIS 2005, Lecture Notes in Computer Science, vol. 3733, 2005.
[39] J. Reichardt, S. Bornholdt, Statistical mechanics of community detection, Phys. Rev. E 74 (2006) 016110.
[40] U.N. Raghavan, R. Albert, S. Kumara, Near linear time algorithm to detect community structures in large-scale networks, Phys. Rev. E 76 (2007) 036106.
[41] D. Holten, Hierarchical edge bundles: visualization of adjacency relations in hierarchical data, IEEE Trans. Vis. Comput. Graph. 12 (5) (2006) 741–748.
[42] D. Holten, J.J. van Wijk, Force-directed edge bundling for graph visualization, Comput. Graph. Forum 28 (3) (2009) 983–990.
[43] W. Cui, H. Zhou, H. Qu, P.C. Wong, X. Li, Geometry-based edge clustering for graph visualization, IEEE Trans. Vis. Comput. Graph. 14 (6) (2008) 1277–1284.
[44] D. Harel, Y. Koren, A fast multi-scale method for drawing large graphs, in: Proceedings of the Working Conference on Advanced Visual Interfaces, AVI '00, 2000, pp. 282–285.
[45] M.A. Smith, B. Shneiderman, N. Milic-Frayling, E. Mendes Rodrigues, V. Barash, C. Dunne, T. Capone, A. Perer, E. Gleave, Analyzing (social media) networks with NodeXL, in: Proceedings of the Fourth International Conference on Communities and Technologies, 2009, pp. 255–264.
[46] N. Henry, J.-D. Fekete, M.J. McGuffin, Nodetrix: a hybrid visualization of social networks, IEEE Trans. Vis. Comput. Graph. 13 (6) (2007) 1302–1309.

Helen Gibson is a researcher at Sheffield Hallam University. She has a BSc in Mathematics from Edinburgh University and studied for her PhD on graph and network visualization at Northumbria University. She is currently working on the Unity and Athena projects which are investigating technological solutions for community policing and disaster management respectively.

Paul Vickers is a UK Chartered Engineer, holds a BSc degree in Computer Studies from Liverpool Polytechnic, and a PhD in Software Engineering & HCI from Loughborough University. He is currently Reader in Computer Science and Computational Perceptualization at Northumbria University where conducts research into visualization and auditory display and their applications.