It's a Wrap: Toroidal Wrapping of Network Visualisations Supports Cluster Understanding Tasks

Kun-Ting Chen
Monash University
Melbourne, VIC, Australia
kun-ting.chen@monash.edu

Tim Dwyer
Monash University
Melbourne, VIC, Australia
tim.Dwyer@monash.edu

Benjamin Bach
University of Edinburgh
Edinburgh, UK
bach@ed.ac.uk

Kim Marriott
Monash University
Melbourne, VIC, Australia
kim.Marriott@monash.edu

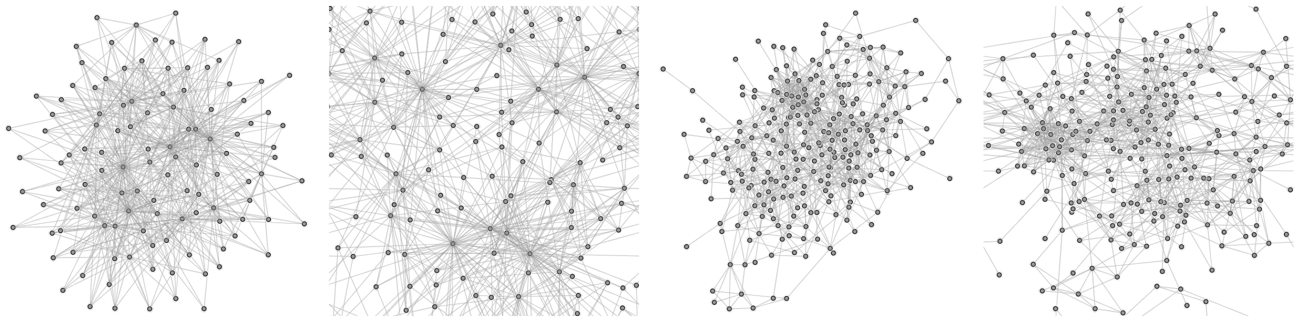

(a) Football-NoTorus (b) Football-Torus (c) Recipe-NoTorus (d) Receipe-Torus

Figure 1: (a-b) shows node-link and toroidal representations of an American College football network (115 nodes, 669 links) [18]. The toroidal layout in (b) provides better use of the available space to provide greater separation between hub nodes. (c-d) shows a recipe network (258 nodes, 1090 links) [2]. The toroidal layout in (d) makes it easier than the node-link representations to identify that there are three main clusters in the network, which supports cluster understanding tasks.

ABSTRACT

We explore network visualisation on a two-dimensional torus topology that continuously wraps when the viewport is panned. That is, links may be "wrapped" across the boundary, allowing additional spreading of node positions to reduce visual clutter. Recent work has investigated such pannable wrapped visualisations, finding them not worse than unwrapped drawings for small networks for path-following tasks. However, they did not evaluate larger networks nor did they consider whether torus-based layout might also better display high-level network structure like clusters. We offer two algorithms for improving toroidal layout that is completely autonomous and automatic panning of the viewport to minimiswe wrapping links. The resulting layouts afford fewer crossings, less stress, and greater cluster separation. In a study of 32 participants comparing performance in cluster understanding tasks, we find that toroidal visualisation offers significant benefits over standard unwrapped visualisation in terms of improvement in error by 62.7% and time by 32.3%.

CHI '21, May 8–13, 2021, Yokohama, Japan
© 2021 Association for Computing Machinery.
ACM ISBN 978-1-4503-8096-6/21/05…$15.00
https://doi.org/10.1145/3411764.3445439

CCS CONCEPTS

• **Human-centered computing** → **Graph drawings**; **Empirical studies in visualization**.

KEYWORDS

Graph visualization; Network visualization; Torus topology; Layout algorithm; Cluster visualisation; User study.

ACM Reference Format:
Kun-Ting Chen, Tim Dwyer, Benjamin Bach, and Kim Marriott. 2021. It's a Wrap: Toroidal Wrapping of Network Visualisations Supports Cluster Understanding Tasks. In *CHI Conference on Human Factors in Computing Systems (CHI '21), May 8–13, 2021, Yokohama, Japan.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3411764.3445439

1 INTRODUCTION

Node-link diagrams are the most common way of visualising network data (or graphs) such as social and trade networks, software architecture, biological networks across many domains. However, as the number of nodes and especially the number of links increases, node-link diagrams become cluttered. In large and dense network diagrams, it becomes difficult both to disambiguate individual links but also to discern the high-level structure of the network, for example, to make out local clusters of high connectivity. For this reason, alternatives to node-link diagrams such as adjacency matrices can be used [3, 26], but these suffer from the disadvantage that they are unfamiliar and less intuitive [30, 32]. Here we explore a different solution that still uses node-link diagrams but provides a less cluttered layout that better reveals the network structure.

Node-link diagrams are traditionally laid out on a flat rectangular viewport corresponding to a printed page or computer display. Highly connected network structures translate into diagrams with many links that tend to cross the diagram leading to a dense, cluttered "hairball" (Fig. 1(a), Fig. 6(b), Fig. 9(a, c)). However, Chen et al. [8] recently evaluated a technique using a force-directed approach for mapping arbitrary network structures embedded on a toroidal surface—a problem long considered by graph theorists—to diagrams with wraparound links in a pannable viewport, see Fig. 2.

Compared to a traditional layout, a toroidal topology diagram allows for links to be wrapped horizontally and vertically around the boundaries of the display, so the layout algorithm has more flexibility to untangle the diagram. Chen et al. demonstrated that for small networks this leads to less link crossings, less variability in link length and greater incidence angle between links where they connect to a node. Through a user study, the authors showed that for small networks and with interactive panning, the resulting layouts supported link following tasks at least as well as traditional layouts. However, there a possible disadvantage of such wrapping is that it requires users to follow wrapped links from one side of the display to the other. Moreover, Chen et al. did not evaluate larger networks nor did they consider whether the additional spreading of torus-based layout might also better display high-level network structure such as clusters.

In this paper, we build on the work by Chen et al. and introduce a new layout method to better visualize cluster structures in toroidal networks. Results show fewer crossings, less stress, greater incidenceangle and greater cluster separation as well as less time and errors in understanding clusters. Our individual contributions are four-fold:

(1) A new algorithm for computing toroidal node-link diagram layout. This algorithm, like Chen et al., is a general-purpose toroidal layout algorithm based on a variant of a force-directed placement. However, unlike Chen et al., our new toroidal layout algorithm is considerably more robust, consistently producing high-quality layouts (see Sect. 4). Our algorithm extends the pairwise gradient descent algorithm of Zheng et al. [47] to handle the more complex case of layout on a torus.
(2) An algorithm for computing how best to "cut" a toroidal network layout to the surface of a torus, i.e., to automatically pan the viewport to reduce the number of links wrapped. To the best of our knowledge this has not been previously considered. As shown in Fig. 2(c) and Fig. 4, our method better reveals clusters and makes a considerable difference to the quality of the final layout.
(3) An evaluation of layout metrics comparing automatic toroidal and non-toroidal layout algorithms using a large corpus of 200 networks. This demonstrates that our introduced torus-based layout algorithm is able to find node positions affording better aesthetics in terms of fewer crossings, less stress, greater incidence angle, and greater cluster distance than either non-torus or Chen et al.'s toroidal layouts.
(4) A user study comparing our new algorithm with automatic panning to traditional network layouts for cluster understanding tasks. This study differs from those in [8] in that we use (a) a new and optimised layout algorithm with (b) automatic wrapping, (c) larger networks (≤ 134 nodes, ≤ 2590 links), and (d) focus on cluster discrimination tasks. We find that torus layout significantly outperformed non-torus for cluster identification tasks in terms of improvement in error by 62.7% and time by 32.3%.

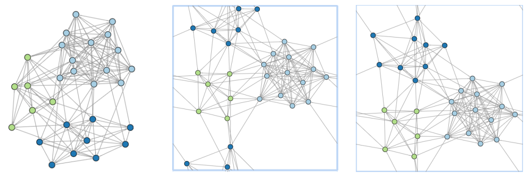

(a) Unwrapped (b) Wrapped (c) Best pan

Figure 2: Example of unwrapped and wrapped networks; Colours are used to illustrate already known clustering information: (a) standard force-directed layout; (b) torus layout wraps the links around the boundaries vertically top-to-bottom, or horizontally left-to-right using our toroidal layout algorithm (Sect. 3); (c) wrapping links are minimised by our automatic pan algorithm (Sect. 3.3)

The full study material, illustrative examples of torus network visualisations based on pairwise gradient descent, and evaluation results are available online: https://observablehq.com/@kun-ting/its-a-wrap.

2 BACKGROUND

Toroidal Layouts: Toroidal embeddings of graphs have long been considered as an interesting problem for study by graph theorists and mathematicians who are interested in their topological properties. These embeddings are of particular interest for these specialists as certain non-planar graphs can be embedded without link crossings on the surface of a torus. Combinatorial algorithms for toroidal embeddings have been developed [23] and adaptations of force-directed layout suggested [22], but until now, to the best of our knowledge, the only attempt to evaluate the utility of toroidal layout for visualisation was reported recently by Chen et al. [8].

Chen et al. [8] evaluate the readability of toroidal embeddings of small graphs (≤ 15 nodes, ≤ 36 links) compared with a traditional non-toroidal view, with and without giving participants the ability to interactively pan the viewport by dragging with the mouse. They tested basic link-following tasks (identify neighbours/find the shortest path) and feature identification tasks (estimate the number of nodes or links). Interactive panning, by allowing users to choose a view that centres the features that they are interested in, turned out to be a key feature. They found that without panning toroidal views were more difficult to use for these tasks than flat (unwrapped) views, but with panning they were similarly readable.

There were two key limitations of the work by Chen et al. First, the small graphs tested are not really representative of the kind of real-world networks of interest in many domains, such as biology. Second, they presented an automatic layout algorithm based on a stress-minimisation approach, but admitted that it could easily become stuck in local minima of the stress function, corresponding to a poor choice of link wrapping across the torus surface. Therefore, the layouts tested in the study involved human-guidance of the algorithm. However, such an interaction would not scale to larger, real-world networks.

Clustered Network Layout: Note that to visualise high-level network structures, there are techniques which explicitly encode groups or clustering information in visualisations [9, 16, 19, 28], or evaluations of task-performance for such group-level visualisations [32, 40, 43]. These algorithms rely on pre-identified cluster information—either from categorical variables within the data or pre-computed community detection—to highlight the cluster boundaries via layout or visual cues. However, we do *not* require knowledge of clusters and we do *not* specifically optimise the graph layout based on clustering information. Instead, our approach, like multidimensional scaling and force-directed graph-layout more generally, implicitly groups nodes by minimising a cost function over difference between ideal graph theoretic distances and actual node separation in the drawing [17].

Automatic Layout Algorithms: Many visualisations rely on solving optimisation problems to generate a readable layout that reveals meaningful patterns in data. For example, force-directed algorithms seek to optimise node-placement to support perception of clusters [13, 14, 27], paths [47], and spatial distances between nodes [7, 10]. The precise goal of the optimisation and the quality of its solution determine how well we can perceive patterns in the resulting visualisation and understand the data. Unfortunately, force-directed placement is a non-trivial optimisation problem, i.e., it is extremely difficult to find a global optimum with respect to the layout goals, and so imperfect heuristics must be used and the results are often sub-optimal. Furthermore, for larger and denser networks even the best force-directed approaches typically produce "hairball" diagrams with little visible structure [30].

Aesthetics Metrics: It is typical in network visualization to evaluate the quality of network layout using metrics that model aesthetic and readability requirements. For example, fewer crossings and greater incidence angle have been shown to improve network path following tasks [21, 35, 37, 46]. We use graph aesthetics measures to evaluate our automatic algorithm, optimizing for cluster discrimination.

3 TWO ALGORITHMS IMPROVING TORUS LAYOUTS

This section introduces our algorithm for drawing an arbitrary network over a 2D torus topology. Similar to Chen et al. [8], our torus-based node-link diagrams are based on the approach of cutting open a torus with two cuts, resulting in a rectangular layout with links wrapping around in north-south and east-west fashion. This topology gives the layout algorithm additional options to route links across the boundaries, reducing crossings, providing more uniform link length and increasing node-link angular resolution. We introduce a toroidal layout algorithm that solves a key limitation of the toroidal layout algorithm proposed in Chen et al. [8] which could become stuck in local minima (as shown in Fig. 6(a)) and thus which often required manual intervention to guide the algorithm to find a reasonable layout. Our new layout algorithm is completely autonomous. We use this algorithm to find layouts that consistently yield better graph aesthetics: fewer crossings, less stress, greater incidence angle, and greater cluster distance (Sect. 4). To better discriminate clusters, we also propose a novel algorithm - finding the best pan. These two algorithms improve toroidal layouts such that they have a better separation of clusters, thereby supports cluster understanding tasks identified through our empirical analysis (Sect. 4) and controlled user study (Sect. 5).

Following Chen et al. [8], our layout algorithm finds node positions of each pair of nodes in a 3×3 repeated tiling. We develop an iterative algorithm which seeks to minimise *stress* of the layout across the surface of the torus. However, while Chen et al. move all pairs of vertices at each iteration and also need a good initialisation to find a suitable torus wrapping, our approach randomly moves a single pair of vertices at a time. In our imperical testing (see Sect. 4.2), we find that this stochastic approach avoids the algorithm getting stuck prematurely in local minima of the stress function and leads autonomously to a high-quality torus layout without the need of a good initial state.

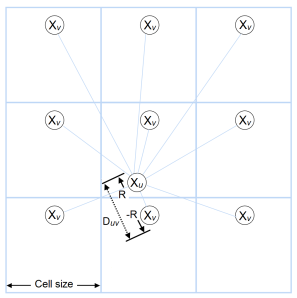
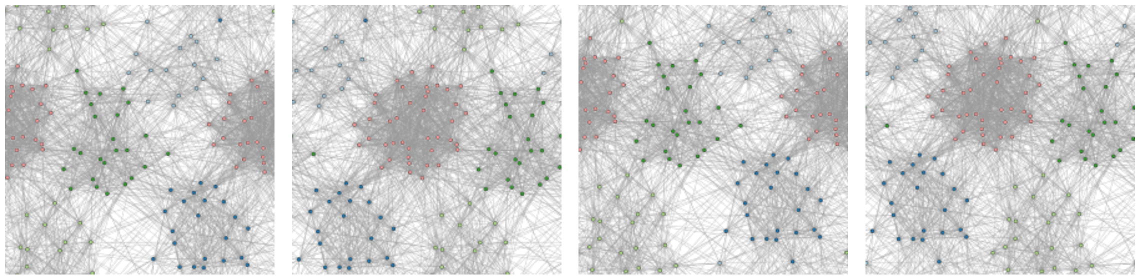


Figure 3: To compute a toroidal layout, we consider gradient descent contribution to the reduction of stress between a node X_u with respect to another node X_v in nine possible adjacencies.

In this paper, we refer to the algorithm of Chen et al. as All-Pairs. Our new algorithm is named Pairwise, given that it minimises the stress function by moving a single pair of nodes at a time using gradient descent. The idea to randomly select and move a single pair of nodes at a time is inspired by an approach suggested for general (non-toroidal) stress-minimising graph layout by Zheng et al. [47].

The key component to adapting a gradient-descent layout approach (such as stress minimisation) to a 2D torus layout, is to consider the nine different possible choices for wrapping each link, as shown in Fig. 3. We find that the Pairwise approach works particularly well for toroidal layout because at each iteration, as well as optimising stress for a single link, we can choose the optimal link wrapping configuration for that link. In the All-Pairs approach,



(a) No pan (b) Best horizontal pan (c) Best vertical pan (d) Best pan in both horizontal and vertical directions

Figure 4: Example of automatic panning of toroidal layouts of a network with 126 nodes and 2496 links from SMALL+HARD (described in Sect. 4.1 and Fig. 7): (a) Original toroidal layout without auto pan requires a user to pan to navigate the network; (b-c) Either horizontal or vertical pan reduces link wrappings across the edges of the display; (d) Best pan in both horizontal and vertical directions minimises the number of link wrappings and better reveals 5 main clusters.

selecting the wrapping configuration for all links at once was the source of significant instability.

3.1 Pairwise Gradient Descent

Force-directed methods are the most commonly-used layout algorithms for general purpose graph visualisation. These methods find a layout by minimising an objective function, such as the standard *stress* function, based on differences between ideal and the actual distances in a low-dimensional coordinate system [7, 10, 11, 47]. We adapt the stress function for 2D toroidally-wrapped topology. We consider the positions of each pair of nodes in a 3×3 repeated tiling, with equal square *cell size* as shown in Fig. 3, which is the length of each square tile. Each node is considered to have nine positions, with the same offset position within every cell. Like Chen et al., for each pair of nodes, we compute the gradient information across the nine possible ways to consider their adjacency. Thus, we have the following definitions for stress for conventional unwrapped graphs NOTORUS, and then for toroidal wrapped graphs TORUS. Given a graph G with nodes V, we define:

$$stress = \begin{cases} \sum_{(u,v)\in V\times V, u\neq v} \frac{(L\times D_{uv}-d_{uv})^2}{(L\times D_{uv})^2} & \text{NoTorus} \\ \sum_{(u,v)\in V\times V, u\neq v} min_{w\in W} \frac{(L\times D_{uv}-d_{uvw})^2}{(L\times D_{uv})^2} & \text{Torus} \end{cases} \quad (1)$$

The constant L is selected proportionally to *cell size* and the graph diameter, i.e., the longest of the shortest-paths of G, as defined in Algorithm 1. For a pair of nodes (u, v), d_{uv} is the Euclidean distance between u and v, D_{uv} is the shortest graph-theoretic path length between them. In TORUS, $w \in \{1 \ldots 9\}$ selects one of the nine possible adjacencies which informs the wrapping as described previously, such that each d_{uvw} is the actual (Euclidean) distance of (u, v) between the centre cell and adjacent cell w. For both NOTORUS and TORUS, the term $\frac{1}{L\times D_{uv}^2}$ is used to penalise long-range attraction.

A gradient-descent approach to reducing the stress function uses the gradient information for this function in a given graph configuration to choose descent vectors, or directions by which to move the nodes to reduce the overall stress function, as per [14]. For TORUS, for a given pair of nodes we choose the adjacency across cells which contributes to the greatest reduction in stress as the descent vector by which the nodes will be moved. In Chen et al. such descent vectors were computed simultaneously across all pairs of nodes, before moving all nodes according to the computed vectors. In this paper, however, we move just a pair of nodes at a time and follow an annealing schedule to enforce convergence.

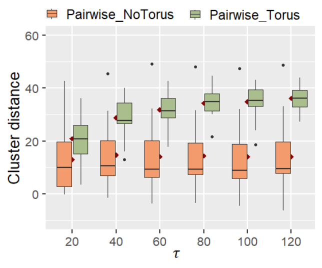
A summary of our annealing schedule is shown in Equation 2, where $\eta(t)$ is a time-dependent scale factor applied to descent vectors before moving nodes accordingly.

$$\eta(t) = \begin{cases} min\ (1, \frac{D_{max}^2}{D_{uv}^2} e^{-\lambda t}) & \text{for} \quad t \leq \tau \\ min\ (1, \frac{D_{min}^2}{D_{uv}^2} \frac{1}{1+\lambda t}) & \text{for} \quad \tau < t \leq \tau_{max} \end{cases} \quad (2)$$

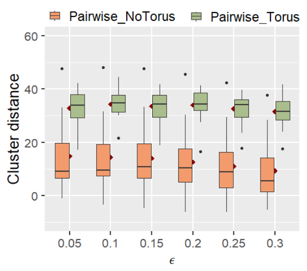
We follow [47] in choosing an exponential decay schedule for a fixed number of iterations τ. Starting from the next iteration after τ, the algorithm uses a $\frac{1}{t}$ schedule to converge to a stable configuration. The annealing schedule begins with a maximum step size, 1 at first iteration $t = 0$. This avoids local minima, as reported in [47]. λ is a decay constant determined by a given parameter ϵ such that the schedule's step size at iteration τ is constrained from above by ϵ, i.e., $D_{max}^2 e^{-\lambda\tau} = D_{min}^2 \epsilon$.

The parameter that determines how well a toroidal layout spreads out clusters of a network is (1) τ which controls when to switch from exponential schedule to the $\frac{1}{t}$ schedule (Equation 2), (2) ϵ which determines the constraint of $\eta(t)$. We experimented with a variety of parameter τ and ϵ for the step size schedule $\eta(t)$ as shown in Fig. 5 and section 1 of the supplementary file. For τ, we fixed $\epsilon = 0.1$ while varying τ in a range of 20 and 120. We used the graphs from SMALL-HARD graph corpus described in Sect. 4.1 and Fig. 7. We find increasing τ led to better separation of clusters and less stress, as shown in Fig. 5(a, c). For ϵ, we then fixed $\tau = 80$ while varying ϵ from 0.05 to 0.3. The results indicate that τ greater than 0.2 is a poor choice and thus it has worse cluster separation and stress, as shown in Fig. 5(b, d). We therefore set $\tau = 80$ and $\epsilon = 0.1$ for the exponential schedule. For $\frac{1}{t}$ schedule we set $\epsilon = 0.001$. This gives smaller step size and thus the layout algorithm always converges.

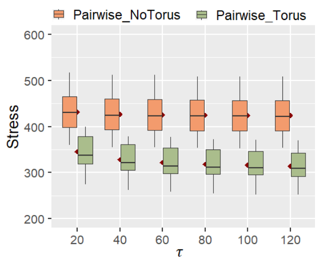
The stopping criterion is set to a maximum pairwise movement $\delta < 0.03$ or a maximum of $\tau_{max} = 200$ iterations , whichever comes



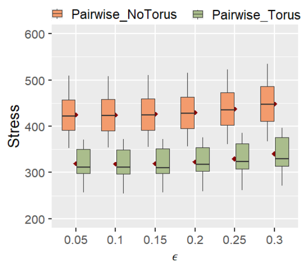
(a)



(b)



(c)



(d)

Figure 5: The cluster distance and stress of networks from SMALL+HARD (Sect. 4.1) with 20 runs when varying parameters τ and ϵ on Equation 2. A larger value of τ gave greater cluster distance (a) and less stress (c) with $\epsilon = 0.1$. There was not much improvement for either $\tau > 80$ (a,c) or $\epsilon > 0.2$ (b,d). Therefore we chose $\tau = 80$ and $\epsilon = 0.1$ for our PAIRWISE layout algorithm.

first. Algorithm 1 gives pseudocode for this process [1]. Fig. 6(b,d) show layout examples of PAIRWISE-NOTORUS and TORUS at convergence.

```
Data: graph $G = (V, E)$, wrapping $W = 3 \times 3$ tiles
Result: Graph embeddings on a 2D plane
$L \leftarrow \frac{Cell\ Size}{min(Graph\ Diameter\ ,\ 2) + 1}$;
$X \leftarrow |V| \times |V| \times 2$ node position matrix;
$D \leftarrow ShortestPathsMatrix(G)$;
for $\eta$ in each annealing schedule in Equation 2 do
    for each $u, v \in V$ in random order do
        if wrapped then
            $X'_{uv} \leftarrow$ set of 9 possible vectors from $u$ to $v$
              across the cells of $W$;
            $d_{uv} \leftarrow$ euclidean lengths of each $X'_{uv}$;
            $R \leftarrow (min_{w \in W} \frac{(L \times D_{uv} - (d_{uvw})^2}{2}) \frac{\overrightarrow{X'_{uvw}}}{d_{uvw}}$;
        else
            $R \leftarrow \frac{(L \times D_{uv} - d_{uv})^2}{2} \frac{\overrightarrow{X_{uv}}}{d_{uv}}$;
        end
        $X_u \leftarrow X_u - \eta R$;
        $X_v \leftarrow X_v + \eta R$;
        if wrapped then translate $X_u, X_v$ back to centre cell;
    end
end
```

Algorithm 1: PAIRWISE gradient descent layout algorithm, which provides a TORUS layout when *wrapped* is true, or a NOTORUS layout when *wrapped* is false.

3.2 Computational Complexity

The time complexity of each iteration of PAIRWISE and ALL-PAIRS stress minimisation is $O(|V|^2)$ where V is a set of vertices. There is a constant factor W due to torus adjacencies which does make our reported run-times uniformly slower than non-torus layout. The space complexity of all the methods are the same, i.e., $O(|V|^2)$ PAIRWISE-TORUS's asymptotic computation time complexity is the same as traditional force-directed layout so we would expect similar scalability (subject to the constant factor W). Further improvement may be possible using multilevel spatial decomposition methods, e.g. [38, 44].

3.3 Automatic panning

As investigated in [8], additional context and interactive panning have been shown to improve understanding of wrapped visualisations. The qualitative feedback in that study revealed that repetition of networks in adjacent cells is not practical for displaying large graphs. Rather, interactive panning was found to be highly-beneficial in helping study participants to pan, such that links of interest are not wrapped. However, we can greatly support this process by automatically panning to minimise the number of links that are split across viewport boundaries.

To perform automatic panning of a given layout (e.g. as determined by the algorithm given in Sect. 3.1, we take a sweepline approach [41] to search for pan positions which lead to the least number and severity of split links, horizontally and vertically. A given layout provides a fixed set of node positions relative to a viewport. A "pan" of a TORUS layout, involves translating all node positions uniformly, except where a node would move outside the viewport, it is wrapped to the other side of the viewport, top-to-bottom or left-to-right. For a given layout of a graph with $|V|$ nodes, the node positions left-to-right define an ordering over the nodes, and the positions top-to-bottom provide a second ordering. The set of links that are wrapped across viewport boundaries $E_{wr} \subseteq E$ is constant under translation of the nodes, until the translation is sufficiently large that a node must be wrapped around. Thus, in each axis (horizontal and vertical) there are precisely $|V|$ distinct translations that must be considered in order to examine all sets of possible wrapped links for a given layout. We can examine each of these sets to determine which induces the lowest *wrapping cost*, defined as 3, where d_{uv} is the Euclidean distance between u and v connected by a link e:

$$wrapcost(E_{wr}) = \sum_{e \in E_{wr}} \frac{1}{d_{uv}} \quad (3)$$

[1]The detailed pseudocode to our method is available from https://github.com/Kun-Ting/its-a-wrap

Algorithm 2 details the steps of this procedure. The runtime complexity of Automatic Panning is O(|V|log|V|+|E|) due to the need to sort node positions and examine all links.

```
Data: Node position vector $X$ with position $(x, y) \in X$ for each
      node, for a torus layout of a graph $G$ from Algorithm 1.
      Fig. 4(a) shows an example.
Result: Torus layout with minimum link wrappings on the
        boundary and less disconnected clusters centred within the
        viewport
Horizontal Sweep (result shown in Fig. 4(b)):
(1) Sort nodes by increasing $(x, \_) \in X$ and initialise sweep-line
    positions $S$ with the mid-points $s_i$ of all adjacent $x_i, x_{(i+1)}$;
(2) For each sweep line position $s_i \in S$, maintain a set of open links.
    Assuming we sweep left-to-right, at each $s_i$ we add to $E_{wr}$ the
    links outgoing from the right side of node $i$ and remove any links
    incoming to the left-side of $i$. If $wrapcost(E_{wr})$ (Equation 3) of
    open links is smaller than $minCost$, take this as the new
    $minCost$ and set $minX$ to the current sweep line position $s_i$;
Vertical Sweep (result shown in Fig. 4(c)):
(3) Repeat step 1 and 2 for all $y$-positions $(\_, y) \in X$. Therefore,
    $minY$ is the sweep line position with minimum cost.
Apply optimal pan and centre within viewport
(result shown in Fig. 4(d)):
(4) Translate all node positions based on $minX$, $minY$.
(5) Centre the layout such that the centre $x$-position of the left-most
    and the right-most nodes is at the centre of the cell; and similar to
    centre vertically.
```

Algorithm 2: Auto Panning Procedure

4 ALGORITHM EVALUATION

In this section, we compare two layout conditions: torus (Torus) and traditional 2D planes (NoTorus) for both our proposed algorithm Pairwise (Algorithm 1) and the All-Pairs algorithm used in [8] against a large corpus of 200 graphs. We show that Pairwise has convergence and run-time performance benefits over its predecessor All-Pairs. We assess layout quality using established graph aesthetics measures and a novel cluster readability metric, *cluster distance*, measuring visual distance between boundaries of clusters in a given layout, finding that Pairwise toroidal layout algorithm outperforms either Pairwise non-torus or All-Pairs layout algorithms.

In order to evaluate how well a layout method separates nodes and clusters, we look at graphs with community structures [29, 39]. We control for *modularity*, a metric for graph theoretic community structure [31]. A graph with high modularity indicates that the cluster structure is more distinct, as there are more links within each cluster than between clusters [15, 31].

We compared means of runtime, stress, crossings, incidence angle, and cluster distance of four different layout methods, i.e., Pairwise-Torus, Pairwise-NoTorus, All-Pairs-Torus and All-Pairs-NoTorus, using Friedman's non-parametric test and Nemenyi's post-hoc pairwise comparison, as they did *not* follow normal distribution. We report significant differences under 95% confidence.

4.1 Graph Corpus

The graphs in our sample corpus were generated using algorithms designed to simulate real-world community structures in graphs [6, 15], using generators from NetworkX [34]. We generated 200 graphs, grouped by two variables: graph modularity [31] (5 levels from low to high: **0.25, 0.3, 0.35, 0.4, 0.45**) and graph size (2 levels: Small: 68-80 nodes, 710-925 links, 3-8 clusters, and Large: 126-134 nodes, 2310-2590 links, 3-8 clusters). This gives us 10 classes. For all classes, the graph density is calculated by the ratio of the number links of the graph to the maximum number of possible links ($\frac{2\times|E|}{|V|\times(|V|-1)}$). This density is fixed at a range of 0.3±0.01.

We use a standard *Random Partition Network* model [15] and *Gaussian Random Partition* model [6] to generate our graph corpus. This gives us graphs with clustering information based on the desired range of modularity, density and size. We exclude graphs whose minimum modularity of an individual cluster is ≤ 0.23, which we found was the minimum to provide visible community structure [15].

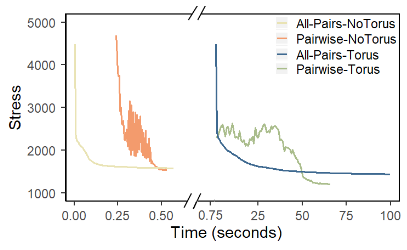
4.2 Runtime comparison

We compare runtime performance of Pairwise algorithm with All-Pairs [8]. We implemented the Pairwise algorithm in JavaScript and D3 [5], and implemented All-Pairs as per [8] in *WebCola*[1]. We ran both algorithms with networks from our graph corpus as described in Sect. 4.1. For each graph, we generated network layouts with 20 random initial node positions controlled by seed within a 1×1 square at the centre for each graph. The configuration of unit link length, stopping criteria, and maximum number of iterations are same for each technique and described in Sect. 3. We record the time of each technique using Google Chrome browser (version 80), running on an Intel i7-7800X (3.5GHz) CPU and 32GB of RAM. Overall, Fig. 7 indicates that Pairwise-Torusis significantly faster (p=0.037) to converge than All-Pairs-Torus by 62% for Small+Easy. Graphics of statistical results and boxplots of runtime comparison are available in Sect. 2.2 of the supplementary file.

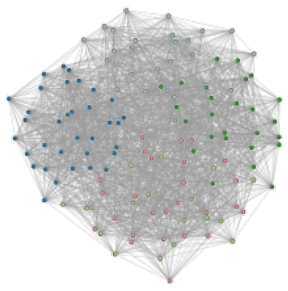
Fig.6(a) shows mean stress over time of an example graph for Large+Hard with 20 runs. An example graph for Large+Easy is in Sect. 2.1 of the supplementary file. The result shows that Pairwise avoids the algorithm getting stuck in local minima of the stress function as opposed to All-Pairs. Furthermore, Pairwise-Torus reaches lower stress levels than All-Pairs-Torus at convergence. While the convergence time is affected by stopping conditions, in our evaluation, we used the same convergence threshold and maximum number of iterations for all methods. However, step size attenuation also affects convergence time. It is computed differently in All-Pairs torus, which is based on gradient contribution of all pairs nodes as in [14] and Pairwise, which we chose 80 iterations as the threshold τ (Equation 2) between exponential and convergence schedule based on experimental results. Experimentally, we found a larger τ led to longer time to converge, but it did not give much improvement in terms of cluster separation, as shown in Fig. 5 and Sect. 1 of the supplementary file. Fig. 6(b-d) show Pairwise-Torus generates higher-quality layouts than either Pairwise-NoTorus or All-Pairs methods at convergence.

4.3 Quality comparison

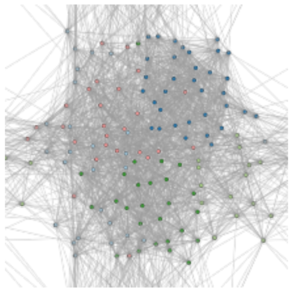
Following Chen et al. [8] we first used a set of layout aesthetic quality metrics to compare NoTorus and Torus using Pairwise



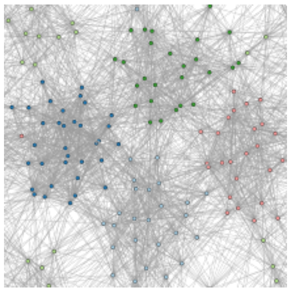
(a) Stress Against Time



(b) Pairwise NoTorus



(c) All-Pairs Torus

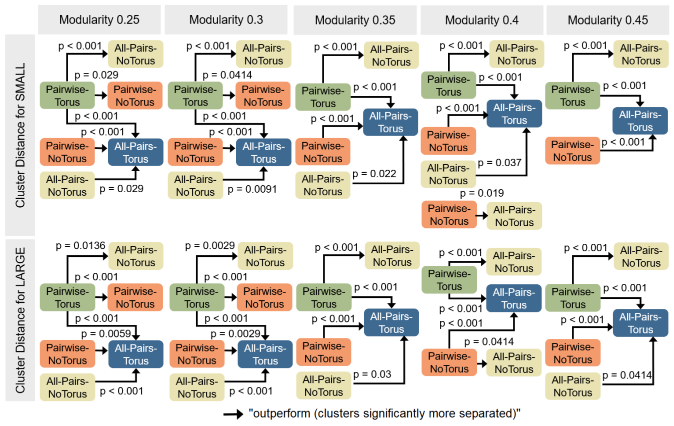


(d) Pairwise Torus

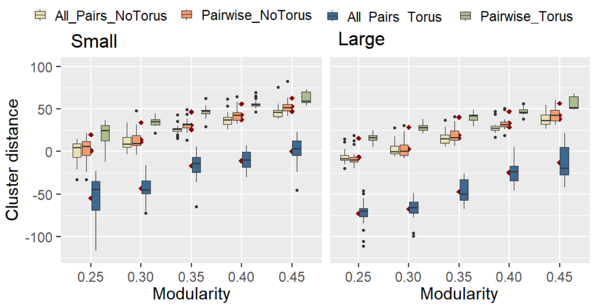
Figure 6: (a) Mean stress over time of 20 runs of torus and standard node-link for both Pairwise and All-Pairs algorithms for a network with 130 nodes, 2504 links from Large+Hard (Sect. 4.1). (b-d) show the network layouts at convergence. Pairwise-Torus reached lower stress level, was faster to converge, and better revealed network clusters than All-Pairs-Torus.

Graph Set	Number Graphs	Nodes	Links	Number Clusters	Graph Density	Topology	Layout Method	Cluster Distance	Stress	Time (sec)	Crossings	Incidence Angle
Clustered Small Modularity=0.4 (SMALL-EASY)	20	72.1	785.1	4.6	0.3	No Torus	All Pairs	37.7	399	0.16	24702	0.957
						No Torus	Pairwise	42.9	393.9	0.08	24376	0.958
						Torus	All Pairs	-10.7	386	22.4	20635	0.951
						Torus	Pairwise	56.1	274.3	8.5	14075	0.946
Clustered Large Modularity=0.4 (LARGE-EASY)	20	129.1	2470.3	4.4	0.3	No Torus	All Pairs	29	1412.2	0.5	277356	0.978
						No Torus	Pairwise	33.7	1408.9	0.3	276040	0.979
						Torus	All Pairs	-24.4	1328.5	85.3	209443	0.975
						Torus	Pairwise	47	1047.1	64.5	158202	0.975
Clustered Small Modularity=0.3 (SMALL-HARD)	20	71.5	774.6	4.7	0.3	No Torus	All Pairs	11.1	427.9	0.16	29555	0.959
						No Torus	Pairwise	14.3	424.6	0.08	29343	0.959
						Torus	All Pairs	-43	422.9	23.2	23984	0.952
						Torus	Pairwise	34.3	318.1	8.4	14751	0.944
Clustered Large Modularity=0.3 (LARGE-HARD)	20	129.1	2535.4	5.3	0.3	No Torus	All Pairs	3.1	1545.4	0.5	363685	0.979
						No Torus	Pairwise	3.3	1543.4	0.3	361353	0.98
						Torus	All Pairs	-67.4	1529.1	86.4	268336	0.975
						Torus	Pairwise	28.6	1220.7	65.6	179452	0.973

Figure 7: Average properties of 80 random partition networks and graph aesthetics results of 20 runs of each network rendered using Torus, NoTorus for both Pairwise and All-Pairs algorithms when varying modularity and size.



(a) Statistical results



(b) Cluster distance

Figure 8: Statistical results and box plots of cluster distance of 200 networks laid out using Torus and NoTorus for both Pairwise and All-Pairs algorithms when varying graph modularity between 0.25 and 0.45 for Small and Large networks.

and All-Pairs. However, we are more interested in whether the additional spreading afforded by torus-based layout is better at showing clusters and so revealing the network structure. This was not considered by Chen et al [8].

4.3.1 Established graph aesthetics metrics . The metrics— *stress*, *minimum incidence angle*, and *number of line crossings*—have been shown important for user performance in readability tasks in the past [36, 45]. Graphics of statistical results and boxplots of stress, incidence angle, and crossings are summarised in Sect. 2.3 of the supplementary file.

Stress—measures how well the layout captures the structure in the underlying network [10]. Graphs with lower stress have been found to be more preferred by users [12]. We calculate stress as per Equation 1. As shown in Fig. 6 and Fig. 7, Pairwise-Torus achieves significantly lower stress ($p<0.001$) than Pairwise-NoTorus, All-Pairs-NoTorus for Small+Easy, Large+Easy, Small+Hard, and Large+Hard. Pairwise-Torus achieves significantly lower stress ($p<0.001$) than All-Pairs-Torus for Small+Easy, Small+Hard, and Large+Hard. For unwrapped layout, we found Pairwise-NoTorus achieves significantly lower stress ($p=0.0135$) than All-Pairs-NoTorus for Small+Easy.

Incidence Angle—Maximising the minimum angle of incidence between links entering a node gives better readability of network connectivity. Following Purchase [36] our metric for incidence angle measures deviation of minimum-incidence angle for each node from the ideal maximum for that node's degree $\theta_v = 360° / degree(v)$. Smaller deviation from this ideal is better. Fig. 7 shows that Pairwise-Torus achieves significantly smaller deviation ($p<0.001$) than Pairwise-NoTorus for Small+Easy, Large+Easy, Small+Hard, and Large+Hard. Pairwise-Torus achieves significantly smaller deviation than All-Pairs-NoTorus for Small+Easy ($p<0.001$), Large+Easy ($p=0.006$), Small+Hard ($p<0.001$) , and Large+Hard ($p<0.001$).

$$\frac{1}{|V|} \sum_{v \in V} \frac{|\theta_v - min\theta_v|}{\theta_v} \tag{4}$$

Line Crossings—The negative effect of line crossings on readability of graphs is well studied in [20, 36]. Fig. 7 shows that Pairwise-Torus achieves significantly ($p<0.001$) fewer crossings than Pairwise-NoTorus and All-Pairs-NoTorus for Small+Easy, Large+Easy, Small+Hard, and Large+Hard. Furthermore, Pairwise-Torus significantly ($p=0.0366$) achieves fewer crossings than All-Pairs-Torus for Small+Easy.

In accord with Chen et al. [8]'s findings for small networks, our results show that torus-based layout have clear benefits over traditional node link diagrams for larger networks at least for these metrics.

4.3.2 New cluster readability metrics—cluster distance. We use a new metric for cluster readability: *cluster distance* to measure how well a layout algorithm is able to separate clusters. For a given layout, for all pairs of clusters whose convex hulls are not overlapping we compute:

Minimum separation between convex hulls—measures space between non-overlapping clusters. A larger value indicates more distance between the boundaries of clusters in a given layout. For torus, we first identify a convex polygon in a 3×3 torus coordinate

	WrapCost		*Wrapping Links*		*Time (sec)*
Auto Pan	*No Pan*	*Best Pan*	*No Pan*	*Best Pan*	*Best Pan*
Small+Easy	4.64	2.41	522.7	324.7	0.04
Small+Hard	4.28	2.49	514.5	354	0.04

Table 1: Automatic panning results: mean wrapCost, number of wrapping links across the boundary and running time of 20 random runs for Small graphs at high (0.4) and low (0.3) modularity (with layout pre-computed by Algorithm 1); graph metrics as shown in Fig. 7.

for each cluster. We then use the Gilbert–Johnson–Keerthi (GJK) algorithm [33] to determine the minimum distance between convex polygons.

Then, for all pairs of clusters that are overlapping, we compute:
Minimum penetration depth between convex hulls—To measure the minimum translation vector required to separate the convex hulls of the cluster pair, as a measure of cluster overlap. A larger value indicates more severely overlapping clusters. We use the Expanding Polytope Algorithm (EPA) which is based on Minkowski sum to compute the penetration depth [42].

Then, for each pair of clusters we define *cluster distance* as the negative minimum penetration depth if they are overlapping, or the minimum separation between convex hulls if they are not overlapping. Across all pairs we take the average minimum cluster distance as a metric of cluster separatedness across the whole graph.

Graphics of statistical results and box plots of cluster distance are summarised in Fig. 8. Overall, we found:

- Pairwise-Torus significantly outperformed All-Pairs- NoTorus, All-Pairs-Torus in cluster distance for both low and high modularity at 0.25, 0.3, 0.35, and 0.4 for both Small and Large.
- Pairwise-Torus outperformed Pairwise-NoTorus in cluster distance for modularity at 0.25, 0.3 for Small and Large.
- We also found Pairwise-NoTorus significantly outperformed All-Pairs-NoTorus for cluster distance under modularity at 0.4 for Small and Large.
- As Pairwise significantly outperformed All-Pairs, we used Pairwise to perform user evaluation for NoTorus and Torus in the next section.

4.4 Automatic panning results

We conducted a small empirical analysis of the Auto Pan Algorithm 2 in terms of number of wrapped links. Results are summarised in Table 1. Automatic panning significantly improves the number of wrapped links, while the *wrapcost* penalty prefers wrapping long links over short links, which tends to keep clusters unwrapped. The difference is visible in Figure 4.

5 USER EVALUATION

The previous section demonstrated the improvement provided by our new algorithm Pairwise with automatic panning, in terms of graph aesthetics and cluster readability metrics. In this section, we investigate if our torus drawings with automatic panning are more effective for people to use than standard unwrapped representations for high-level network topology analysis tasks.

Discerning high-level structure, such as clusters, is an important task in many domains, e.g., community structure in social networks[31], self-organizing maps of documents[24]. Visual cluster analysis is an important application of network visualisation. For example, an analyst may need to visually verify whether the output of an automatic cluster labelling algorithm makes sense with respect to the graph structure. For this task, they want a layout algorithm that displays the connectivity structure of the graph as clearly as possible. However, clutter makes cluster disambiguation difficult with standard node-link diagrams. The hypothesis that we test in our user study is whether the additional spreading afforded by toroidal layout in terms of the new cluster separation metric (Sect. 4.3.2) also leads to better human perception of clusters. We do not use the All-pairs-Torus algorithm [8] in the user study because the results of the empirical experiments (Fig. 7, Fig. 8) overwhelmingly demonstrate that the new Pairwise algorithm is superior.

The particular task that we focus on in this paper is the inspection of community structure, i.e., cluster identification. This study is different from Chen et al.'s study with small graphs (≤ 15 nodes, ≤ 36 links) [8]. They did not test whether torus-based layout might also better display high-level network structure like clusters. Our study graphs (Sect. 5.3) are 4.7 to 8.6 times larger in the number of nodes, 21 to 68 times larger in the number of links than Chen et al. In summary, we are interested in answering the following research question:

RQ: Do toroidal layouts with interactive wrapping provide more benefit to perception than a standard unwrapped representation for *cluster identification*?

5.1 Techniques & Setup

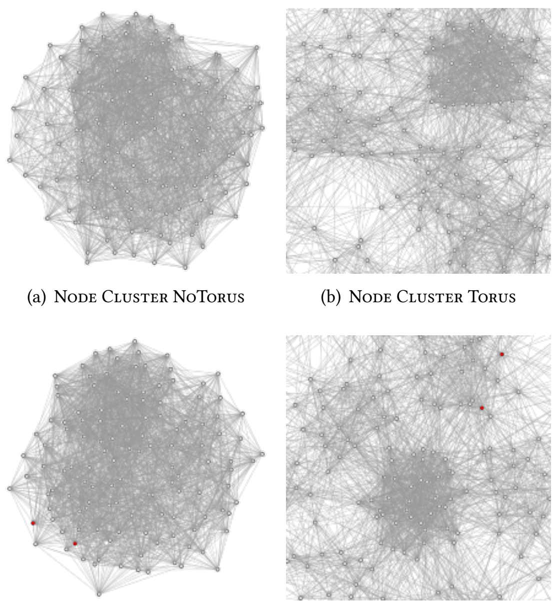
The techniques in our study are NoTorus and Torus generated using our new Pairwise algorithm as detailed in Sect. 3. In the study, each trial starts with automatic panning for Torus. We did not show any cluster colours in this study to not hint towards any graph structure. Interactive panning using the mouse is enabled only in the Torus condition. A user can drag the visualisation such that when the view is panned off one side of the display, it reappears on the opposite side.

5.2 Tasks & Dependent Variables

For our study, we selected two representative network visualisation tasks, inspired by existing task taxonomies [25, 39], and that involve the understanding of clusters. For each task, we record task-completion time (Time), task-error (Error), and subjective user confidence in using for the task in terms of rank (1 or 2) (Confidence). We record pan distance as the overall distance the participant moves the mouse for each trial. We record user ranking for learnability for cluster related tasks in general (Learnability) and overall preference (Overall).

Our tasks are as follows:

- **Cluster Number: Identify the number of clusters (Fig. 9(a, b))** Participants are required to count the number of clusters they can find in the image. Participants are provided radio buttons to answer 1 to 10. We calculate Error as the absolute difference between the correct answer and the user's response, divided by the correct answer.
- **Node Cluster: Do the two red nodes belong to the same cluster (Fig. 9(c, d))?** We record participants' responses through multiple-choice questions with the values *yes*, *no*, and *not sure* as options. Error is binary with not-sure counting as error.


(a) Node Cluster NoTorus (b) Node Cluster Torus

(c) Node Cluster NoTorus (d) Node Cluster Torus

Figure 9: Example of Cluster Number tasks (a,b) and Node Cluster tasks (c,d) for Large+Hard: Modularity 0.3, 130 nodes, 2575 links, 5 clusters (a,b); Modularity 0.3, 126 nodes, 2496 links, 5 clusters (c,d)

5.3 Graph Structure

To evaluate effectiveness of toroidal drawings for cluster identification tasks, we used graphs from our graph corpus in Sect. 4.1, with two levels of difficulty (Easy: modularity=0.4, Hard: modularity=0.3) and two levels of graph size (Small and Large), whose graph metrics are summarised in Fig. 7. We name these four groups: Small+Easy, Large+Easy, Small+Hard, and Large+Hard. In each group, we randomly selected 5 graphs for trials for each task. The number of clusters for Cluster Number ranged between 4 and 7. For Node Cluster, the number of clusters ranged between 5 and 7, as we found in pilot studies that fewer than five clusters is too easy for Node Cluster. We ran the Pairwise layout algorithm 20 times for a chosen graph for both NoTorus and Torus. We then selected one layout at random and used it for both NoTorus and Torus for each study trial.

5.4 Hypotheses

Our hypotheses were pre-registered with the Open Science Foundation: https://osf.io/7vbr4.

For Cluster Number:

- **H1**: Torus has better task effectiveness (in terms of time and error) than NoTorus independent of difficulty levels.
- **H2**: For Easy graphs, NoTorus has better task effectiveness (in terms of time and error) than Torus (requires certain panning and mental wrapping).
- **H3**: For Hard graphs, Torus has better task effectiveness (in terms of time and error) than NoTorus.
- **H4**: Participants will report more confidence in Torus than NoTorus.

For Node Cluster:

- **H5**: Torus has better task effectiveness (in terms of time and error) than NoTorus independent of difficulty levels.
- **H6**: Torus has better task effectiveness (in terms of time and error) than NoTorus for both Easy and Hard tasks.
- **H7**: Participants will report more confidence in using Torus than NoTorus.

For participant preference

- **P1**: Overall, participants prefer Torus over NoTorus.

5.5 Experimental Design

We use a within subject design with 2 techniques (Torus, NoTorus) × 2 tasks (Cluster Number, Node Cluster) × 2 level of difficulty (Easy, Hard) × 2 sizes (Small, Large) × 5 recorded repeats. This leaves us with a total of 80 recorded trials per participant. We blocked the study by tasks, i.e., participants would do both techniques with the same task before moving to the second task. For each task block, we counterbalanced the order of the techniques using a full-factorial design. The order of each level of difficulty and size in each technique was the same: Small+Easy→Large+Easy→ Small+Hard→Large+Hard. The order of trials for each technique within each level was randomised.

5.6 Participants and Procedures

We recruited 32 participants from local institutes through university's email list and snowballing. 19 were males, 13 were females. The age of participants was between 20 and 50 (mean = 31.5). 22 people reported seldom or never using network diagrams while 10 people often used network diagrams in their work or study.

While run entirely online due to COVID-19 health-concerns, the experimenter supervised each participant through remote video conferencing software to give proper instructions, assure the participants' engagement, and help with eventual questions. The experimental software was loaded in a participant's Google Chrome browser. Each participant shared their screen with the experimenter. The experimenter ensured that each participant used a monitor with resolution no less than 1366 ×768 pixel. Each study trial used a stimuli with a size of 650 × 650 pixels. Each trial was correctly loaded in a participant's browser, before the recording started. The experimenter trained each participant to identify a cluster as a graph structure whose links within a cluster are relatively more than the links between clusters. Before each new task and technique, there were 2 training trials. When a participant gave an answer in these training trials, an image with different clusters highlighted in different colours appeared, outlining the cluster. Participants first had to complete all training trials correctly before proceeding to the recorded trials. Each recorded trial had a timeout of 20 seconds to prevent participants from trying to perform precise link counting. For Torus technique, short animations demonstrating interactive torus wrapping were shown.

5.7 Results

All of the participants completed the training and recorded trials. Therefore we recorded performance for 2,560 trials. Since Error was not normally distributed, we used Friedman's non-parametric test and Tukey's posthoc multiple pairwise comparison to identify significant differences between NoTorus and Torus. The residuals of Time were normally distributed, visually checked with Q-Q plots and a histogram, supported by a Shapiro-Wilk test. The variances of the Time were equal by Levene's test. We therefore performed a 3-way repeated measures ANOVA to test significant difference in Time between NoTorus and Torus. We used paired t-test for posthoc pairwise comparisons with p-value adjustment using Bonferroni correction. Wilcoxon's non-parametric signed rank test was used to analyse paired significances of subjective user rank. Confidence intervals indicate 95% confidence for mean values for all the pairwise comparisons.

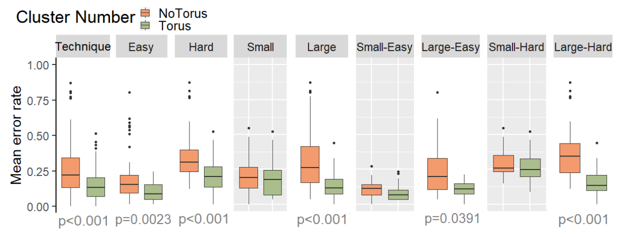
Overall, for Cluster Number, we found the following significant results:

- For Error, we found Torus significantly outperforming NoTorus for technique (M(NoTorus)=.26, M(Torus)=.15)), Easy (M(NoTorus)=.18, M(Torus)=.09)), Hard (M(NoTorus)=.33, M(Torus)=.2)), Large (M(NoTorus)=.31, M(Torus)=.12)), Large+Easy (M(NoTorus)=.25, M(Torus)=.1)) and Large+Hard (M(NoTorus)=.37, M(Torus)=.15)), as shown in Fig.10(a).
- For Time, we found NoTorus (M=13.7s) significantly outperforming Torus (M=15.1s) for only Hard (Fig.10(c)).
- Participants report significantly more confidence in using Torus than NoTorus (p<0.001) (Fig. 11).

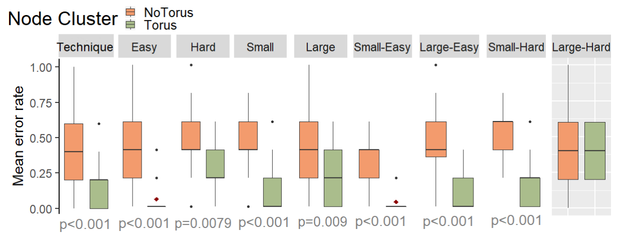
For Node Cluster, we found the following significant results:

- For Error, Torus significantly outperformed NoTorus for technique (M(NoTorus) = .43, M(Torus) = .16), Easy (M(NoTorus) = .39, M(Torus) = .05), Hard (M(NoTorus) = .47, M(Torus) = .28), Small (M(NoTorus) = .44, M(Torus) = .09), Large (M(NoTorus) = .42, M(Torus) = .23), Small+Easy (M(NoTorus) = .33, M(Torus) = .03), Large+Easy (M(NoTorus) = .45, M(Torus) = .06) and Small+Hard (M(NoTorus) = .55, M(Torus) = .16), as shown in Fig. 10(b).
- Torus significantly outperformed NoTorus in time for Easy (M(NoTorus) = 9.84s, M(Torus) = 7.22s) and Large+Easy (M(NoTorus) = 10.5s, M(Torus) = 7.1s)), as shown in Fig.10(d).
- Participants report significantly more confidence in using Torus than NoTorus (p<0.001), as shown in Fig. 11.

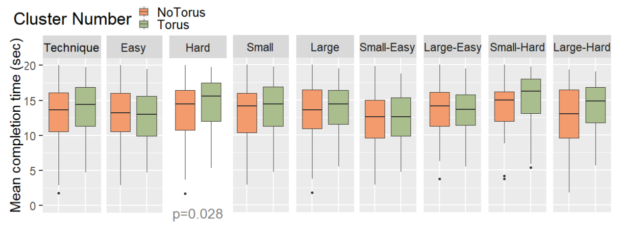
Overall, Torus improved task effectiveness in terms of error rate by 42.3% on average, compared with NoTorus for Cluster Number task. For Node Cluster task, Torus improved the task effectiveness in terms of error rate by 62.7% and time by 32.3% on average, as opposed to NoTorus. Participants reported that Torus is significantly easier to learn (Learnability) (p<0.001) than NoTorus for our cluster related tasks. Torus was significantly preferred (Overall) (p<0.001) over NoTorus in overall rank, as shown in Fig. 11. Based



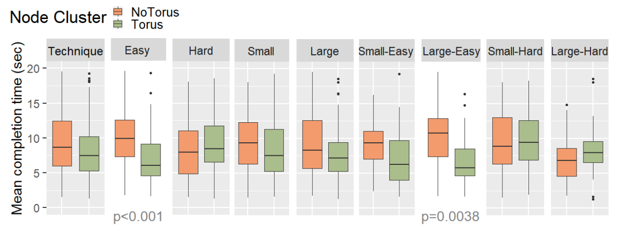
(a) Cluster Number error



(b) Node Cluster error



(c) Cluster Number time



(d) Node Cluster time

Figure 10: User evaluation results of mean error and time between NoTorus and Torus split by task, difficulty and size. Columns of significant results are shown in white background.

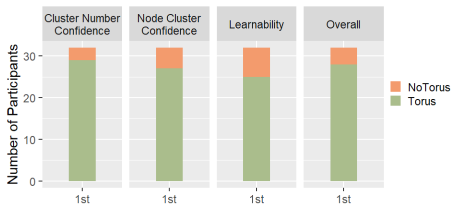


Figure 11: Subjective user rank of NoTorus and Torus

on these results, for **RQ** we rejected hypothesis H2 and accepted H4, H7, P1. We partially accepted H1, H3, H5 for error, and H6 for error and time for Easy.

5.8 Qualitative User Feedback

For understanding clusters, the majority of participants mentioned that the Torus with panning provided for less overlap between nodes and was better spreading nodes and clusters more spatially, thus reducing visual clutter and increasing readability. While our automatic panning optimises the view a user sees first, interaction further helps to find the best view possible. Some participants reported that Cluster Number required more panning to identify the graph structure. For Node Cluster, on the other side, participants reported that they rarely used interactive panning, except for verifying their answer. This is supported by our measurements that Node Cluster, required less user interaction (majority less than 1000 pixels per trial). The box plot can be found in Sect. 3.3 of the supplementary file. There are 4 participants who reported they favoured NoTorus as it gives a full overview of the networks and did not require to identify the same piece of cluster wrapped around top-bottom or left-right.

5.9 Discussion

Our results indicate that NoTorus is sometimes faster than Torus when counting the number of clusters for HARD (Fig. 10(c)), but the error rate in the NoTorus condition was generally much higher (Fig. 10(a)). We believe participants were sometimes faster with NoTorus because they were simply guessing the answer. Our new study is intended to be complementary to Chen et al. [8] by evaluating an important task not considered in their study. Since our new layout algorithm produces high-quality torus layout with fewer crossings, larger incidence angles and less stress, we would expect that the results from Chen et al. for low-level connectivity understanding and path following tasks would also be reproducible for larger networks using our new algorithm. However, we leave such an evaluation to future work.

6 CONCLUSION AND FUTURE WORK

Chen et al. [8] demonstrated that when compared to traditional layouts, torus-based node-link diagram layouts reduce the number of link crossings, reduce stress and increase incidence angle of links entering a node. However, their user study found no benefits of torus-based layouts for detailed network analysis tasks with small graphs.

We have presented a new algorithm for torus-based layout of networks that is more robust than the previous method of Chen et al. [8]. Furthermore, we improve the resulting layout by using a novel algorithm that minimises the number of wrappings across the boundary by centring elements of interest in the display.

Using this algorithm, we have investigated whether toroidal wrapped layouts provide advantages over traditional network layouts for the higher-level task of understanding network structure. Both an analysis of graph metrics and a user study have clearly shown that they do: participants were able to more accurately determine the number of clusters in a graph and more quickly and accurately determine whether two nodes are in the same cluster with the toroidal wrapped layouts. Furthermore, participants preferred the toroidal wrapped layouts. This is the first demonstration that torus-based layouts provide real-world benefits over traditional node-link layouts.

Our research also reveals directions for future research. While our layout algorithm converges significantly more quickly than that of Chen et al. [8], the need to compare 9 different alternatives for link wrappings at each iteration means that it is still considerably slower than the corresponding layout algorithms for traditional node-link diagram layout. Improving its speed is a major direction for future research.

Another future research direction is to investigate the use of toroidal wrapped layouts to understand high-dimensional data when using multi-dimensional scaling (MDS) [4]. Stress, as considered in Equation 1 is also used in MDS and the similarity between MDS and graph layout algorithms has been observed before [17]. It seems very likely that toroidal wrapped layouts will also better show clusters and high-level structure in MDS visualisations.

REFERENCES

[1] 2015. WebCoLa: Constraint-Based Layout in the Browser. https://ialab.it.monash.edu/webcola

[2] Yong-Yeol Ahn, Sebastian E Ahnert, James P Bagrow, and Albert-László Barabási. 2011. Flavor network and the principles of food pairing. *Scientific reports* 1 (2011), 196.

[3] Michael Behrisch, Benjamin Bach, Nathalie Henry Riche, Tobias Schreck, and Jean-Daniel Fekete. 2016. Matrix reordering methods for table and network visualization. In *Computer Graphics Forum*, Vol. 35. Wiley Online Library, 693–716.

[4] Ingwer Borg and Patrick JF Groenen. 2005. *Modern multidimensional scaling: Theory and applications.* Springer Science & Business Media.

[5] Mike Bostok. 2011 (accessed 21 April, 2020). *https://d3js.org/.* https://d3js.org/

[6] Ulrik Brandes, Marco Gaertler, and Dorothea Wagner. 2003. Experiments on graph clustering algorithms. In *European Symposium on Algorithms.* Springer, 568–579.

[7] Ulrik Brandes and Christian Pich. 2008. An experimental study on distance-based graph drawing. In *International Symposium on Graph Drawing.* Springer, 218–229.

[8] Kun-Ting Chen, Tim Dwyer, Kim Marriott, and Benjamin Bach. 2020. DoughNets: Visualising Networks Using Torus Wrapping. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems.* 1–11.

[9] Christopher Collins, Gerald Penn, and Sheelagh Carpendale. 2009. Bubble sets: Revealing set relations with isocontours over existing visualizations. *IEEE Transactions on Visualization and Computer Graphics* 15, 6 (2009), 1009–1016.

[10] Sabin Devkota, Reyan Ahmed, Felice De Luca, Katherine E Isaacs, and Stephen Kobourov. 2019. Stress-Plus-X (SPX) Graph Layout. In *International Symposium on Graph Drawing and Network Visualization.* Springer, 291–304.

[11] Tim Dwyer. 2009. Scalable, versatile and simple constrained graph layout. In *Computer Graphics Forum*, Vol. 28. Wiley Online Library, 991–998.

[12] Tim Dwyer, Bongshin Lee, Danyel Fisher, Kori Inkpen Quinn, Petra Isenberg, George Robertson, and Chris North. 2009. A comparison of user-generated and automatic graph layouts. *IEEE transactions on visualization and computer graphics* 15, 6 (2009), 961–968.

[13] Tim Dwyer, Kim Marriott, Falk Schreiber, Peter Stuckey, Michael Woodward, and Michael Wybrow. 2008. Exploration of networks using overview+ detail with constraint-based cooperative layout. *IEEE Transactions on Visualization and Computer Graphics* 14, 6 (2008), 1293–1300.

[14] Tim Dwyer, Kim Marriott, and Michael Wybrow. 2008. Topology preserving constrained graph layout. In *International Symposium on Graph Drawing.* Springer, 230–241.

[15] Santo Fortunato. 2010. Community detection in graphs. *Physics reports* 486, 3-5 (2010), 75–174.

[16] Emden R Gansner, Yifan Hu, and Stephen G Kobourov. 2009. Gmap: Drawing graphs as maps. In *International Symposium on Graph Drawing.* Springer, 405–407.

[17] Emden R Gansner, Yehuda Koren, and Stephen North. 2004. Graph drawing by stress majorization. In *International Symposium on Graph Drawing.* Springer, 239–250.

[18] Michelle Girvan and Mark EJ Newman. 2002. Community structure in social and biological networks. *Proceedings of the national academy of sciences* 99, 12 (2002), 7821–7826.

[19] Yifan Hu, Emden R Gansner, and Stephen Kobourov. 2010. Visualizing graphs and clusters as maps. *IEEE Computer Graphics and Applications* 30, 6 (2010), 54–66.

[20] Weidong Huang, Peter Eades, and Seok-Hee Hong. 2009. Measuring effectiveness of graph visualizations: A cognitive load perspective. *Information Visualization* 8, 3 (2009), 139–152.

[21] Weidong Huang, Seok-Hee Hong, and Peter Eades. 2008. Effects of crossing angles. In *2008 IEEE Pacific Visualization Symposium.* IEEE, 41–46.

[22] Stephen G Kobourov and Kevin Wampler. 2005. Non-Euclidean spring embedders. *IEEE Transactions on Visualization and Computer Graphics* 11, 6 (2005), 757–767.

[23] William Kocay, Daniel Neilson, and Ryan Szypowski. 2001. Drawing graphs on the torus. *Ars Combinatoria* 59, 2 (2001), 259–277.

[24] Teuvo Kohonen. 1982. Self-organized formation of topologically correct feature maps. *Biological cybernetics* 43, 1 (1982), 59–69.

[25] Bongshin Lee, Catherine Plaisant, Cynthia Sims Parr, Jean-Daniel Fekete, and Nathalie Henry. 2006. Task Taxonomy for Graph Visualization. In *Proceedings of the 2006 AVI Workshop on BEyond Time and Errors: Novel Evaluation Methods for Information Visualization* (Venice, Italy) *(BELIV '06).* ACM, New York, NY, USA, 1–5. https://doi.org/10.1145/1168149.1168168

[26] Shixia Liu, Weiwei Cui, Yingcai Wu, and Mengchen Liu. 2014. A survey on information visualization: recent advances and challenges. *The Visual Computer* 30, 12 (2014), 1373–1393.

[27] Jiawei Lu and Yain-Whar Si. 2020. Clustering-based force-directed algorithms for 3D graph visualization. *The Journal of Supercomputing* (2020), 1–62.

[28] Wouter Meulemans, Nathalie Henry Riche, Bettina Speckmann, Basak Alper, and Tim Dwyer. 2013. Kelpfusion: A hybrid set visualization technique. *IEEE transactions on visualization and computer graphics* 19, 11 (2013), 1846–1858.

[29] Nina Mishra, Robert Schreiber, Isabelle Stanton, and Robert E Tarjan. 2007. Clustering social networks. In *International Workshop on Algorithms and Models for the Web-Graph.* Springer, 56–67.

[30] Tamara Munzner. 2014. *Visualization analysis and design.* CRC press.

[31] Mark EJ Newman. 2006. Modularity and community structure in networks. *Proceedings of the national academy of sciences* 103, 23 (2006), 8577–8582.

[32] Mershack Okoe, Radu Jianu, and Stephen Kobourov. 2018. Node-link or adjacency matrices: Old question, new insights. *IEEE transactions on visualization and computer graphics* 25, 10 (2018), 2940–2952.

[33] Chong Jin Ong and Elmer G Gilbert. 1997. The Gilbert-Johnson-Keerthi distance algorithm: A fast version for incremental motions. In *Proceedings of International Conference on Robotics and Automation*, Vol. 2. IEEE, 1183–1189.

[34] Python Package. 2019. NetworkX graph library. https://networkx.github.io. Last accessd: September thirteenth, 2020.

[35] Helen Purchase. 1997. Which aesthetic has the greatest effect on human understanding?. In *International Symposium on Graph Drawing.* Springer, 248–261.

[36] Helen C Purchase. 2002. Metrics for graph drawing aesthetics. *Journal of Visual Languages & Computing* 13, 5 (2002), 501–516.

[37] Helen C Purchase, David Carrington, and Jo-Anne Allder. 2002. Empirical evaluation of aesthetics-based graph layout. *Empirical Software Engineering* 7, 3 (2002), 233–255.

[38] Aaron Quigley and Peter Eades. 2000. Fade: Graph drawing, clustering, and visual abstraction. In *International Symposium on Graph Drawing.* Springer, 197–210.

[39] Bahador Saket, Paolo Simonetto, and Stephen Kobourov. 2014. Group-level graph visualization taxonomy. *arXiv preprint arXiv:1403.7421* (2014).

[40] Bahador Saket, Paolo Simonetto, Stephen Kobourov, and Katy Börner. 2014. Node, node-link, and node-link-group diagrams: An evaluation. *IEEE Transactions on Visualization and Computer Graphics* 20, 12 (2014), 2231–2240.

[41] Michael Ian Shamos and Dan Hoey. 1976. Geometric intersection problems. In *17th Annual Symposium on Foundations of Computer Science (sfcs 1976).* IEEE, 208–215.

[42] Gino Van Den Bergen. 2001. Proximity queries and penetration depth computation on 3d game objects. In *Game developers conference*, Vol. 170.

[43] Corinna Vehlow, Fabian Beck, and Daniel Weiskopf. 2017. Visualizing group structures in graphs: A survey. In *Computer Graphics Forum*, Vol. 36. Wiley Online Library, 201–225.

[44] Chris Walshaw. 2000. A multilevel algorithm for force-directed graph drawing. In *International Symposium on Graph Drawing.* Springer, 171–182.

[45] Colin Ware, Helen Purchase, Linda Colpoys, and Matthew McGill. 2002. Cognitive measurements of graph aesthetics. *Information Visualization* 1, 2 (2002), 103–110.

[46] Vahan Yoghourdjian, Yalong Yang, Tim Dwyer, Lee Lawrence, Michael Wybrow, and Kim Marriott. 2020. Scalability of Network Visualisation from a Cognitive Load Perspective. *arXiv preprint arXiv:2008.07944* (2020).

[47] Jonathan X Zheng, Samraat Pawar, and Dan FM Goodman. 2018. Graph drawing by stochastic gradient descent. *IEEE transactions on visualization and computer graphics* 25, 9 (2018), 2738–2748.