Network Communities of Dynamical Influence

Ruaridh Clark\textsuperscript{1,*}, Giuliano Punzo\textsuperscript{2}, and Malcolm Macdonald\textsuperscript{1}

\textsuperscript{1}Department of Mechanical and Aerospace Engineering, University of Strathclyde, Glasgow, United Kingdom
\textsuperscript{2}Department of Automatic Control and Systems, University of Sheffield, Sheffield, United Kingdom
\textsuperscript{*}ruaridh.clark@strath.ac.uk

ABSTRACT

Fuelled by a desire for greater connectivity, networked systems now pervade our society at an unprecedented level that will affect it in ways we do not yet understand. Nature, in contrast, has already developed efficient and resilient large-scale networks, including brain connectomes and bird flocks. These natural systems rely on the stimulation of key elements, which access effective pathways of communication, to instigate response and consensus. In this paper, we explore the link between network structure and dynamical influence to further our understanding of these effective networks. Our technique identifies key vertices that rapidly drive a network to consensus, and the communities that form under their dynamical influence, by investigating the relationships between the system’s dominant eigenvectors. These Communities of Dynamical Influence enable the clear identification of human subjects from their brain connectomes and provides an insight into functional activity. They are also used to highlight the effectiveness of starling flocks, where increasing the outdegree is likely to produce a less responsive flock that has the most influential birds poorly positioned to observe a predator and, hence, instigate an evasion manoeuvre.

Introduction

Human capacity to create networked systems is expanding with the growing popularity of cloud services\textsuperscript{1}, improvements in mobile (cellular) network infrastructure and the expansion of the internet, including satellite based provision\textsuperscript{2}. Increased connectivity is not without its drawbacks where denial of service attacks, exploiting Internet of Things (IoT) devices, are an example of the vulnerabilities that can emerge in large, complicated, networks\textsuperscript{3}. To insure against our growing reliance on networked systems, this paper strives to further our understanding of how network topology influences the dynamical response and function of large-scale systems.

Artificial networks are sophisticated but they cannot compete with nature’s accomplishments, where the human brain is estimated to contain 100 billion neurons and 100 trillion synaptic connections\textsuperscript{4}. It is not currently possible to capture, in detail and at such a large scale, this neuronal activity. Instead brain scans can be translated into networks on the scale of millions of vertices\textsuperscript{5}. For the far smaller neuronal system of the \textit{Caenorhabditis elegans}\textsuperscript{6} and a low resolution network representation of the macaque connectome\textsuperscript{7}, the flow of information has been simulated to reveal the existence of bottlenecks and clusters. Such numerical models of information flow become intractable at a sufficiently large scale, therefore this paper uses a spectral
method to uncover communication dynamics in some of nature’s most sophisticated networks, including 1.8 million vertex human connectomes. Insights are gained from these systems by detecting the communities that form due to dynamical influence, where each is led by one of the network’s most influential vertices.

Community Detection

Common approaches for community detection, partitioning and graph clustering often consider either the edge direction or the graph spectral properties, such as modularity, random walk based methods and spectral clustering. Modularity detects communities by comparing the density of edges present with the density that would be present if all the edges were reassigned a new destination vertex at random. Random walk based methods can either emulate the properties of the first eigenvector of the adjacency matrix, see PageRank, or be used to identify communities from modelled diffusion of information. Two commonly used spectral clustering approaches are normalised cuts and k-means clustering. The approach, detailed herein, is similar to that of k-means clustering where both consider multiple eigenvectors of the Laplacian matrix to define communities. In the case of k-means clustering, it employs a heuristic algorithm (k-means) to separate the network into k clusters where each cluster is defined in relation to one of k centroids. These centroids iteratively adjust their positions with the final composition of the clusters dependent on the initial placement of the centroids. There is no optimal solution to the selection of k for any given network instead multiple runs can be performed to determine the sensitivity of k to a chosen criteria.

In this paper, we detect communities without the use of recursive optimisation, instead the number of communities, their members, and influential vertices are all a product of the network’s spectral properties. The communities, defined herein, are referred to as Communities of Dynamical Influence (CDI) as each is associated with the dynamical influence of an influential vertex. These vertices are detected with the first left eigenvector of the Laplacian matrix, which has been employed previously to identify the most effective vertices for spreading information or disease in a dynamical network. The first eigenvector of the adjacency matrix is also widely used as a centrality metric, where it identifies vertices that would be visited most often by agents performing a random walk on the graph. The most common approaches for defining influential communities combine community detection, using modularity or k-core, in conjunction with a function to maximise the influence of vertices assigned to a single community. In these methods, each vertex is given an influence value that is defined either independently of the network topology, by applying a weighting to the vertices, or as a product of the topology, by using Katz centrality (a form of eigenvector centrality). Influence can also be assessed after community designation, where an influence propagation model can be used to assess each community’s influential reach. In this paper, we do not attempt to maximise the influence of community members but instead consider the influence that propagates from vertices. Information propagation of linear consensus has been used previously to define influential communities, where each community is composed of vertices that are on one of the most effective pathways from an influential vertex. In this paper, the same definition is applied for community detection but we do not rely on a model or numerical process, instead communities and dynamical influence can be detected from the relationships between the system’s eigenvectors.
Influence of Network Perturbations

Information propagation can be captured analytically for consensus processes where the first eigenvalue of the system matrix is used to assess the speed of consensus\(^{22}\). Optimising consensus speed by applying an effective input perturbation provides a useful validation of the CDI and their network influence. Similar perturbation optimisations have been studied extensively in the context of leadership selection for the control of multi-agent and swarm systems\(^{22-26}\). In these cases, the perturbation is often constrained so that only a set number of leaders are chosen with a binary option for perturbation input, i.e. vertices set as leaders or followers. In the context of multi-agent systems, there is significantly more literature on minimising the steady-state variance about an input perturbation\(^{23, 24}\) than there is on fast convergence to consensus. There has been an attempt to tackle both problems, but this work was restricted to 1-D community and ring graphs\(^{25}\), and an examination of how the proportion of leader vertices affects the convergence rate to consensus in multiplex networks\(^{26}\). Of most relevance, to the work herein, is globally bounded input perturbations that can be applied with a variable distribution to any combination of vertices\(^{22}\). For such a case, the first left eigenvector of the Laplacian matrix was identified as a sub-optimal resource allocation (equivalent to an input perturbation) for achieving fast convergence to consensus\(^{22}\). An improvement in this allocation has since been developed for directed \(k\)-outdegree graphs, where a near-optimal perturbation vector can be produced by combining first left eigenvectors from manipulated versions of the adjacency matrix\(^ {27}\). It is worth noting that the globally bounded perturbation optimisation problem, to maximise convergence rate to consensus, has no verifiable solution. A numerical optimiser can produce near-optimal solutions, but detection of the global minima is not guaranteed. A significant contribution of this article is the filtering out of local minima by highlighting the most effective network influencers, a process that functions at any network scale and is no longer restricted to \(k\)-outdegree graphs.

By considering how perturbations influence linear consensus, researchers were able to model a starling flock in the presence of white noise\(^ {28}\). They demonstrated that starlings employ an optimal outdegree (six to seven for all flock members\(^ {29}\)), which maximises robustness and allows them to manage uncertainty in consensus\(^ {28}\). The current paper builds on this work by considering how the outdegree, maintained by starlings, can facilitate their responsiveness to perturbation triggering predators.

Results

The Communities of Dynamical Influence (CDI) are detected using the first three left eigenvectors of the Laplacian matrix, as described in the Methods section. CDI are found for a \(k = 5\) nearest neighbour (\(k\)-NNR) network with 50 vertices, in Figure 1, where the relationship between the first three eigenvectors reveals three communities that are each aligned with an influential vertex. The number of CDI is entirely dependent on the number of influential vertices. These vertices are defined as not having an outward (observing) link to a vertex with a larger position vector with respect to the origin, i.e. a vertex that is further from the origin of the eigenvector coordinate system than any of its connections. The first left eigenvector, \(v_{L1}\), value of each influential vertex captures its ability to lead the whole network and, therefore, reflects the relative influence of each community.
Figure 1. The eigenvector values for a directed, 50 vertex, $k$-NNR toy example where $k = 5$ are shown in a where the axes are the first and second left eigenvector and b for the second and third left eigenvector. The network is partitioned into Communities of Dynamical Influence (CDI) with the influential vertices in each community highlighted by a red outline.

Brain Similarity Detection

The detection of CDI in connectomes helps to demonstrate that these communities are found consistently as long as the network is not dramatically altered. The CDI are identified for a series of connectomes, generated by Roncal et al.\(^5\), to recognise a subject’s brain by comparing the influential communities present in scans conducted at different times. These connectomes were created from magnetic resonance imaging (MRI) scans carried out by Landman et al.\(^30\). Each graph contains 1,827,240 vertices that each represent a 1 mm\(^3\) volume of the brain. The centre of each volume (voxel) forms a three dimensional grid with 1 mm spacing between neighbouring voxels. Each edge in the graph is defined as any two vertices that are connected by at least a single fibre, where an edge of weight 1 would represent a single fibre connection. This results in an undirected network of weighted edges.

Landman et al. used 21 healthy volunteers, where each subject was scanned twice with a short break between scan and rescan (scan 1 and scan 2 respectively)\(^30\). For each subject, the CDI for large graphs procedure was applied to the adjacency matrices of both scan and rescan data, as detailed in the Methods section. These CDI are visualised in Fig. 2, for the scan and rescan of a subject, where the CDI for large graphs procedure applies a threshold of 0.01 when assigning vertices to a community (i.e. $(v_A)_i > 0.01$ where $v_A$ represents any of the three eigenvectors used in the CDI coordinate system). Note that one of the scans, for subject 127, could not be sourced and, hence, this article shall consider the remaining 20 volunteers.

Subject Comparison

In the work by Roncal et al.\(^5\), the Frobenius Norm was successfully used to detect the similarity of the scan-rescan matrices from Landman et al.’s study\(^30\). The Frobenius Norm is an established matrix distance measure\(^31\), referred to as Frobenius Distance when assessing graph similarity. The result from Roncal et al.’s study\(^5\) has proven to not be exactly reproducible with the published dataset\(^30\), where the diagonal (scan-rescan comparison) entries do not always produce the lowest values (i.e.
Figure 2. Influential communities from the first and second scan of subject 742 using the CDI for large graphs procedure with the first three eigenvectors of the adjacency matrix. The influential vertex is highlighted for each community. 

(a) Posterior angled view where three outlines are generated by taking slices of a brain surface model, (b) right lateral view, (c) posterior view, (d) superior view.

The similarity of graphs is assessed by comparing the communities detected using a metric, introduced in the Methods section as the mean number of matching communities. Fig. 3a employs the CDI for large graphs procedure to identify communities using the first three eigenvectors, those associated with the largest eigenvalues of the adjacency matrix. For some subjects in Fig. 3b the diagonal entries clearly show the largest number of matching communities, with some of these strong matches reflected in the Frobenius Distance results in Fig. 3a. However, for other subjects the mean number of matching communities is as little as 1, for the diagonal entries in Fig. 3b. In Fig. 3c and d, where a broader range of influential communities are considered, all
Figure 3. Comparison of scan 1 and 2 for Landman et al.\textsuperscript{30} subjects using a Frobenius Distance b,c, d Mean number of matching communities with the CDI for large graphs. In b the CDI for large graphs are defined using the first three eigenvectors, c also includes communities from the 4\textsuperscript{th}, 5\textsuperscript{th} and 6\textsuperscript{th} eigenvectors, d also includes communities from the 7\textsuperscript{th}, 8\textsuperscript{th} and 9\textsuperscript{th} eigenvectors of the adjacency matrix.

diagonal entries indicate a clear match. This indicates that changes in the influence of communities, between scan and rescan, have taken place for those subjects that are not a clear match in Fig. 3 a, but can be clearly identified in b and c. For Fig. 3 c, CDI for large graphs is employed to define communities using the first three eigenvectors and then to assign any vertices without a community to one using the CDI for large graphs procedure for the 4\textsuperscript{th}, 5\textsuperscript{th} and 6\textsuperscript{th} eigenvectors. Fig. 3 d adds further communities by using the CDI for large graphs procedure with the 7\textsuperscript{th}, 8\textsuperscript{th} and 9\textsuperscript{th} eigenvectors. For Fig. 3 c and d the diagonal entries (the scan-rescan comparisons) for all of the subjects produce a notably larger mean number of matching communities than any other entries in their respective rows and columns.
It is worth noting that there are always errors in the images produced from MRI scans, even when using the same equipment and procedure, with small errors occurring because of slight changes in image orientation and magnetic field instability\textsuperscript{32}. The \textit{Mean Number of Matching Communities} is, therefore, designed to accommodate any small positional errors when detecting overlapping communities, as detailed in the Methods Section.

**Validation of Community Influence**

When referring to the influence of a community, we are referring to the influence of the influential vertices in that community. These vertices wield the greatest local influence, but they can often have significant influence beyond that local cluster to the network as a whole. To validate the claim that the CDI are influential, we performed a series of analyses comparing optimised perturbations for driving convergence to linear consensus. The vertices highlighted by this optimisation are those that can lead the network effectively to a new state of consensus, i.e. those with strong local and global network influence, which should align with the influential CDI vertices.

The CDI are used as an input to a perturbation optimiser, detailed in the Methods section (Algorithm 3). The results of this optimisation are compared with the Communities of Influence (CoI) method\textsuperscript{27} and a numerical optimiser\textsuperscript{33} in Fig. 4. The CoI method generates optimised perturbations by detecting influence, using the first left eigenvector, and investigating how this influence changes when key vertices are removed from the network. This was show to be effective in $k$-outdegree networks where the CoI method, using 5 input vectors, produced similar results to the output of a numerical optimiser\textsuperscript{27}. In Fig. 4, the CDI optimiser is shown to produce similar result in $k$-outdegree networks to the CoI method but achieves notably superior results when applied to variable $k$ networks. The CDI-based optimiser never produces a result below 0.9 for the Consensus Speed Ratio, with respect to the numerical optimiser’s result, and sometimes exceeds the result of the numerical optimiser.

The $k$-means clustering algorithm developed by Ng et al.\textsuperscript{34} also employs eigenvectors and a form of machine learning to define clusters. Fig. 5 demonstrates that, for a $k$-NNR graph, the clusters generated by $k$-means clustering (Fig. 5 b) can partition the network in a broadly similar manner to the CDI (Fig. 5 a), as long as the value of $k$ is selected to be the same as the number of CDI. In fact, for the example shown in Fig. 1 the community designation is sometimes identical, with $k$-means clustering prone to variance due to its heuristic nature. An optimised leadership perturbation is defined and overlays the network in Fig. 5 according to the same numerical optimiser as before\textsuperscript{33}. These optimised perturbations appear to form groupings that are matched by the CDI in Fig. 5 a with each of these groupings placed in a separate community. In comparison, $k$-means clustering does not provide such a clear separation in Fig. 5 b with the least influential community (yellow) sprawling across two perturbation groupings.

By using the different community methods as an input to a perturbation optimisation, defined in Algorithm 2 and 3 in the Methods section, the claim that the CDI are better aligned with the optimised perturbation can be further investigated. The results of these optimisations find that the CDI-based perturbation produces a faster convergence speed to consensus ($\lambda = 0.0164$) than the $k$-means clustering communities ($\lambda = 0.0162$).
Responsive Starling Flock Topology

Starlings are an interesting case study for communities of dynamical influence, as they are believed to maintain six to seven outward connections at all times regardless of the flock density\textsuperscript{29}. The reason for this \( k \) neighbour connection rule is unclear, but Cavagna et al. proposed two possible options\textsuperscript{35}. The first suggestion was a cognitive limit; a claim that is supported by experiments on the cognitive limits of pigeons, which were found to only be able to track up to seven objects\textsuperscript{36}. The second proposition was that this number may have evolved to produce the most effective graph for transferring information across the flock. A similar conclusion was reached by research into flock robustness to uncertainty where seven connections per bird was found to produce a more robust flock model in the presence of white noise\textsuperscript{28}. The true reason for this \( k \) neighbour limit is still a matter of debate, but this section will present evidence for why a lower outdegree is beneficial for flocking starlings.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{figure.png}
\caption{Consensus Speed Ratio, with reference to a numerical optimiser, for four topology types with 10 graphs at each vertex size interval. \textbf{a} \( k \)-NNR networks and \textbf{b} Erdős-Rényi random networks with an outdegree set at 10. \textbf{c} \( k \)-NNR networks and \textbf{d} Erdős-Rényi random networks with the outdegree varied between 3 and 10 with connections sampled at random from a uniform distribution.}
\end{figure}
Figure 5. Communities defined by a the CDI method and b k-means clustering for a k-NNR network with 100 vertices, $k = 10$ and all edge weights set at 0.0667. Communities are denoted by colour according to their influence over the whole network. A numerically optimised perturbation overlays the network with circles that are proportional to the perturbation magnitude.

Starling flocks tend towards a thicknesses of between 0.13 and 0.27, where the flock thickness is the ratio of the smallest to largest dimension of an ellipsoid having the same principal moments of inertia as the flock. In Fig. 6, five examples of 1200 bird starling flock networks are presented with a thickness of 0.2, where the flock is modelled by randomly distributing birds from a uniform distribution within a rectangular prism. The community distribution and optimised perturbations are shown for these representative examples that employ $k$-NNR topologies for the following outdegrees; $k = 7, 10, 25, 50$ and 100. The starling vertices remain in the same position for each analysis with the number of nearest neighbours $k$ the only change that affects the network structure.

What is of particular interest, from these results, is the position of the influential vertices in each example. In high $k$ outdegree examples, such as Fig. 6 c, d and e, the influential vertices, associated with the most influential communities, are centrally located. A CDI-based optimised perturbation is shown to align with the most influential communities, therefore the perturbation become increasingly centrally located as the outdegree increases. For the $k = 100$ example, in Fig. 6 e, these perturbations are exclusively located in the centre of the model flock. For lower outdegree examples, such as Fig. 6 a and b, the influential vertices, especially those associated with the most influential communities, are more evenly distributed throughout the flock, aided by the increase in the number of CDI present. Considering an actual starling flock, where $k \approx 7$, the topology makes it more likely that one of the most influential birds will be near the site of a predator attack than in a higher outdegree topology. Therefore, the low outdegree topology is more likely to have a highly influential bird involved in leading a predation avoidance manoeuvre.

The perturbation driven convergence speed to consensus also varies with the outdegree. For the starling flock model, using the same bird distribution each time, a network of 1200 vertices can be generated for every outdegree between $k = 5$ and $k = 50$. The value of the dominant eigenvalue $\lambda$ of the Laplacian matrix, which represents convergence speed, is assessed for each network with the results roughly conforming ($R^2 = 0.962$ ) to a power law distribution ($\lambda = 0.0031k^{-0.184}$). A higher
convergence speed is indicative of a graph that can be more effectively led by key vertices. The highest convergence speeds, therefore, tend to belong to network with lower outdegrees, but for the position distribution used this was not always case. Finally it is worth noting that, although a low outdegree will result in a fast perturbation driven consensus, the outdegree must be large enough to maintain a connected flock. If the outdegree is too low the flock will split whenever a perturbation is applied and will not reconnect.

Defining this lower bound is beyond the scope of the paper but Balister et al. can provide a guide, where connectivity in a $k$-NNR graph can be assured if $k \geq 0.9967 \log N$, where $N$ is the number of vertices. It is worth noting that the starling flock topology is only approximated as a $k$-NNR topology and that the network is constantly changing with birds shifting positions and making new connections.
Discussion

In this article a method for detecting Communities of Dynamical Influence (CDI) is proposed that does not require any heuristic elements. This algorithm identifies communities in a similar manner to some other spectral based methods. In particular, it was shown in some cases to produce a similar result to k-means clustering, but it is notable that CDI can be generated without needing to specify the number of communities. Instead, the number of CDI are a product of the network topology and defined using a coordinate system composed of the first three eigenvectors of the Laplacian. The creation of a perturbation optimiser using CDI, for maximising convergence rate to consensus, demonstrated that the most influential vertices in the network are those that lead influential CDI.

The approach for generating CDI can be applied in the analysis of large graphs. In this article a series of, 1.8 million vertex, human brain networks were analysed to identify influential neuronal communities. The identified communities enabled separate MRI scans to be clearly recognised as belonging to the same subject. The expansion of CDI, to include those defined by the 4th to 9th eigenvectors of the adjacency matrix, indicated that this analysis could highlight changes in neuronal activity while still recognising the same subject. We propose that the subjects with the lowest mean number of matching communities when using CDI with the first three eigenvectors, see Fig. 3 b, are those that display the greatest change in neuronal activity. Whereas a high number of matches in the scan-rescan comparison, when using CDI with the first three eigenvectors, indicates a high similarity in brain activity. This capability could provide quantitative assessment of the change in brain activity over time, if the CDI were monitored for a subject that performed the same activity at different points in time. The emergence of brain stimulation techniques could open avenues to incorporate this knowledge into the treatment of patients, by ensuring key communities (pathways) are strengthened through additional stimulation.

The CDI can be repeatably detected with the number of communities only changing when the topology is altered. For a starling flock model, these insights revealed the benefit to starlings of maintaining a low outdegree. Higher outdegrees were seen to reduce the responsiveness of the network, with the flock becoming composed of fewer CDI. It was also noted that the most influential vertices became located centrally in the flock, where they are unlikely to detect an incoming predator. Such conclusions would be harder to justify using other community detection methods that require a sensitivity analysis or equivalent to determine the optimal number of communities.

Methods

A graph is defined as $G = (V, E)$, where there is a set of $V$ vertices and $E$ edges, which are unordered pairs of elements of $V$ for an undirected graph and ordered pairs for a directed graph.

The adjacency matrix, $A$, is a square $n \times n$ matrix when representing a graph of $n$ vertices. This matrix captures the network’s connections where $a_{ij} > 0$ ($a_{ij}$ is the $ij$th entry of the graph’s adjacency matrix) if there exists a directed edge from vertex $i$ to $j$ and 0 otherwise. Variable edge weights contain information on the relative strength of interactions, whilst uniform edge weighting either only represent the presence of a connection or is a result of all the edges having the same information carrying
capacity. For an undirected graph, the adjacency matrix is symmetric with an edge \((i, j) \in E\) resulting in \(a_{ij} = a_{ji} > 0\). For a directed graph, the indegree is equal to the column sum, \(\sum_i a_{ij}\), and outdegree is equal to the row sum, \(\sum_j a_{ij}\).

The Laplacian matrix is defined as \(L = D - A\) where the degree matrix, \(D\), is a diagonal matrix and the \(i^{th}\) diagonal element is equal to the outdegree of vertex \(i\). The first eigenvalue of the adjacency matrix \((\lambda_1)\), referred to as the Spectral Radius, is the largest eigenvalue in magnitude and is associated with the Perron vector, which is an eigenvector that contains only positive entries. Whereas for the Laplacian matrix the first eigenvalue is associated with the eigenvalue \(\lambda_1 = 0\). For a directed graph, the left eigenvectors of the Laplacian matrix, \(v_L\), are row vectors satisfying \(v_L L = \lambda v_L\).

**Communities of Dynamical Influence**

The Communities of Dynamical Influence (CDI) are found by analysing three left eigenvectors of the Laplacian matrix as presented in Algorithm 1. The algorithm only considers the Real part of any eigenvector. Hence, if the second and third eigenvectors form a complex conjugate pair then the algorithm will use the first, second and fourth eigenvectors since the real part of the complex conjugates will be identical.

Algorithm 1 assesses the coordinates for each vertex as defined by the three eigenvector entries. We consider the most effective community, and network, leaders (influential vertices) to be those vertices that do not "follow" nodes with greater network influence. Therefore, the influential vertices are those with no outward connections to a vertex that is further from the origin, in this eigenvector-based coordinate system. This can be seen in Fig. 1a and b where each community is associated with one of these influential vertices (highlighted by a red outline).

Each vertex is assigned to a community when there is a directed path with vertices in order of influence from lowest to highest, which is the community’s influential vertex as detailed in Algorithm 1. The number of communities is equal to the number of influential vertices detected in the network where a vertex can only belong to one community. If a vertex is assigned to multiple communities, then it is assessed to determine which influential vertex it is in closest alignment with before removing it from all other communities. This alignment is determined by comparing the position vectors of the vertex with respect to the influential vertices.

The proposed approach is described for a single connected component. In the case of a graph with multiple connected components, a separate community detection would need to be performed for each component, as each eigenvector only captures the influence of vertices in one of the components with vertices outwith that component having a null entry.

**CDI for large graphs**

The CDI, reported in Algorithm 1, can be optimised for large graphs to reduce the computational resources required, while also producing additional insights. The procedure for large graphs is altered so that only the most prominent vertices are assigned to each community, where prominence is judged by the magnitude of the eigenvector entries used in the CDI coordinate system. Therefore, a vertex has to exceed a threshold value for at least one of the three eigenvectors, used to define CDI, to be included when assigning communities. A useful by-product of this alteration is that less influential CDI, associated with less dominant
Algorithm 1 Detecting Communities of Dynamical Influence (CDI)

1. Find the first three, normalised, eigenvectors of the Laplacian, $L \in \mathbb{R}^{n \times n}$, $v_{L1}$, $v_{L2}$ and $v_{L3}$.
2. if $v_{L2}$ and $v_{L3}$ are a complex conjugate pair, where $\text{Re}(v_{L2}) = \text{Re}(v_{L3})$ then.
3. $v_{L3}$ is replaced with $v_{L4}$
4. end if
5. Set the position vector $e = [\text{Re}(v_{L1}), \text{Re}(v_{L2}), \text{Re}(v_{L3})]$ and $s = |e|$.
6. Define the influential vertices as any vertex $i \in V$ where $s_i > s_o$ for all $o$ vertices connected to an outward edge from $i$. If $i$ has no outward edges, $s_o = 0$.
7. For each influential vertex $i$ that is found, define a separate community.
8. Populate each community with vertices that can reach the community’s influential vertex $i$ through a sequence of outward edges, where $s_{source} > s_{sink}$ for each edge in the sequence (source and sink denote the vertices at the start and end of the edge, respectively).
9. For any vertex $j$, which is assigned to multiple communities, identify the influential vertex $i$ that produces the largest scalar projection with respect to $j$, i.e. $(e_i \cdot e_j)/s_i$. Remove $j$ from all other communities, where $(e_i \cdot e_j)/s_i < (e_i \cdot e_j)/s_I$ for an influential vertex $i$.
10. Order the communities from largest to smallest $s_i$ value for the influential vertices, to indicate most to least influential communities.

eigenvectors, can now be revealed alongside the most dominant CDI, associated with the first three left eigenvectors. Assuming that there are vertices in the graph with eigenvector entries that do not exceed the threshold value for any of the first three left eigenvectors. In fact, this process can be repeated more than once where any unassigned vertices, which have neither exceeded the threshold value for any of the first three eigenvectors or the proceeding 4th to 6th eigenvectors, are assigned to communities using the CDI for large graphs procedure where the 7th to 9th eigenvectors are used as an input. Such an example of community assignment can be found in Fig. 3.

Matching Brain Communities

For the large brain connectome graphs, the adjacency matrix was employed, rather than the Laplacian, using the CDI for large graphs procedure. This was due to the difficulty that emerged in converging on $\lambda_1 = 0$ for these large matrices that contained more than one near zero eigenvalue. It should be noted that due to the undirected nature of the connectome data, the Laplacian matrix’s ability to highlight the imbalance between outdegree and indegree is less relevant.

The CDI for large graphs procedure is applied with an eigenvector entry threshold of $0.01$ (i.e. $(v_A)_i > 0.01$ where $v_A$ is any of the eigenvectors used in the CDI coordinate system). These communities can then be compared to identify which graphs are generated from scans of the same brain. A key aspect of this comparison is the evaluation of the number of matching communities, where the communities found in each scan 1 graph are compared with those detected in each scan 2 graph. The comparison metric, developed here, considers the shortest distance from all the vertices of one community to the nearest vertex that belongs to another. Vertices were considered overlapping if they are from the same voxel or they are in an adjacent voxel (i.e. maximum overlapping voxel distance $\sqrt{3} \approx 1.74$ mm). The percentage of overlapping vertices are calculated for each community comparison to find the highest percentage overlap between two communities in separate scans. The communities appear to reveal pathways in the brain as depicted in Fig. 2. These pathways can sometimes vary in density of vertices and in length, which makes an exact match between two communities unlikely. Therefore, for a pair of communities to be considered
a match their percentage overlap had to exceed a threshold value. The mean number of matching communities was determined by taking the mean number of matches from a range of threshold values between 50% and 90%, at 10% intervals. Note that any community can only be a member of one matching pair, i.e. if one community in scan 1 overlapped with multiple communities in scan 2 only one of those matching pairs would be considered for the number of matching communities.

**Frobenius Distance**

When applying the Frobenius norm, to assess the difference between two matrices, it is often referred to as the Frobenius Distance and defined as

$$||A||_F = \sqrt{\sum_{i=1}^{n} \sum_{j=1}^{n} |a_{ij} - b_{ij}|}$$  \hspace{1cm} (1)

where $a_{ij}$ and $b_{ij}$ are elements of the adjacency matrix for scan 1 and scan 2 respectively.

**Perturbation Driven Consensus.** The networks considered herein have $N$ agents connected via local communication with a static, time-invariant, topology. The directed graph is at least weakly connected with one giant component (i.e. for every pair of distinct vertices there exists an undirected path). A uniform signal $\mathbf{u} = u[1,1,\cdots,1]^T \in \mathbb{R}^N$ is supplied to all agents with different positive gains $c_i$, where $i = 1,2,\cdots,N$. The dynamics of this system are defined as

$$\dot{x}_i = \sum_{j=1}^{N} a_{ij}(x_j - x_i) + c_i(u - x_i)$$  \hspace{1cm} (2)

where $x_i$ is the state of the $i^{th}$ agent and $u$ is the scalar target value that all agents must achieve. The resource allocation, $c_i$, ranges from 0 to 1, is globally bounded as $\sum c_i = 1$, and scales the comparison between the uniform input signal, $u$, and the current state $x_i$.

The global dynamics of the network can be expressed with respect to the Laplacian matrix as

$$\dot{\mathbf{x}} = -L\mathbf{x} + C(\mathbf{u} - \mathbf{x})$$  \hspace{1cm} (3)

where $C$ is the perturbation matrix, $C = \text{diag}(c) = \text{diag}(c_1,\ldots,c_P)$. Spanning trees have been highlighted previously as a condition for consensus\textsuperscript{38–40} where for a directed network $G$, defined by Eq. (3), consensus will eventually be achieved if all agents are reachable, via directed edges, from the vertices supplied with perturbation input.

**Perturbation Optimisation**

The objective function here is to maximise the system’s convergence to consensus by applying a globally bounded perturbation to the vertices. By changing the coordinates, it has been demonstrated that Eq. 3 can be written as

$$\frac{d\mathbf{y}}{dt} = -(L + C)\mathbf{y}.$$  \hspace{1cm} (4)
where the diagonal elements of $C$ can be optimised to maximise the magnitude of $\lambda_1(-(L+C))$, the rightmost eigenvalue (i.e. eigenvalue with the largest real part) of the negated and perturbed Laplacian matrix.

Only a selection of CDI are used as input vectors for the optimisation, to prevent long computation times when many hundreds of CDI are present in a network. Consensus requires all vertices to reach the same state, therefore the selection may require more isolated communities with strong local influence as well as globally influential communities to produce an optimal perturbation. This selection, detailed in Algorithm 2, took 10 of the most prominent CDI according to the first three left eigenvectors, then included 5 more isolated communities according to the 4th to 6th left eigenvectors and finally added a single community according to only the first left eigenvector. This selection ensures that strong global and local community leaders are present in the optimisation.

**Algorithm 2** Input Vector Selection for Optimisation

1: Detect $m$ Communities of Dynamical Influence (CDI) using Algorithm 1.
2: For each community, create an input vector, $\omega_i$, where $(\omega)_j = |v_{IL}|$ if $j$ is in the community and $(\omega)_j = 0$ otherwise.
3: For each input vector, set $e$ as the vertex with the largest entry in magnitude with the position vector
   
   $$e = [\text{Re}(v_{IL})_e, \text{Re}(v_{L2})_e, \text{Re}(v_{L3})_e],$$

   normalised position vector $n = e / \sqrt{\text{Re}(v_{IL})_e^2 + \text{Re}(v_{L2})_e^2 + \text{Re}(v_{L3})_e^2}$
   and $s = [e]$.
4: If the dot product of any two normalised position vectors is greater than 0.999, these vectors are closely aligned and only the input vector with the largest value of $s$ is considered for selection.
5: For the eligible input vectors, select the vectors associated with the largest 10 values of $s$ in order from largest to smallest. If $m < 10$, then select $m$ input vectors.
6: For each input vector, set $e$ as the vertex with the largest entry in magnitude with the position vector
   
   $$e = [\text{Re}(v_{IL})_e, \text{Re}(v_{L5})_e, \text{Re}(v_{L6})_e],$$

   normalised position vector $n = e / \sqrt{\text{Re}(v_{IL})_e^2 + \text{Re}(v_{L5})_e^2 + \text{Re}(v_{L6})_e^2}$
   and $s = [e]$.
7: If the dot product of any two normalised position vectors is greater than 0.999, these vectors are closely aligned and only the input vector with the largest value of $s$ is considered for selection.
8: For the remaining eligible input vectors, select the vectors associated with the largest 5 values of $s$ in order from largest to smallest adding them onto the existing list of input vectors. If $m < 15$, then select $m - 10$ input vectors.
9: If $m > 15$ then for the remaining eligible input vectors, select the vector associated with the largest value of $|\text{Re}(v_{IL})_e|$ and place it first in the order of input vectors.

For each of the CDI selected by Algorithm 2, an input vector $\omega_i$ is created with entries populated if the vertex is in the community and values set to zero otherwise. The populated entries are given the value corresponding to their entry in the first left eigenvector of the system. These vectors are then manipulated, using the Power Optimisation method described below, and combined to produce the final optimised perturbation, with weighting variables used to determine the ratio of each vector’s contribution.

The Power Optimisation focuses resources on the most effective leaders by raising an eigenvector to a power, $\eta$, for a given input vector, $\omega_i$, according to

$$p_i = \frac{\omega_i^\eta}{\sum_j (\omega_i)_j \eta}$$

(5)

where $\omega_i^\eta$ indicates an element-wise operation and the denominator ensures that $\sum_j (p_i)_j = 1$. When $\eta \to 0$ the vector, $p_i$, approaches a uniform vector state. As $\eta$ is increased, the Power Optimisation method iteratively reduces the value of the
smaller vector elements while increasing the value of the larger elements.

In the following equation $n$ power optimised input vectors, $p_i$, are combined using weighting variables, $r = \{r_1, ..., r_{n-1}\}$, to produce the optimised perturbation vector as follows

$$c = \frac{p_1 + \sum_{j=1}^{n-1} \frac{p_{j+1}}{r_j}}{1 + \sum_{j=1}^{n-1} \frac{1}{r_j}}$$

(6)

where the denominator ensures that $\sum_j(c)_j = 1$. Also note that $C = \text{diag}(c)$ in the $-(L+C)$ system.

Algorithm 3 presents the perturbation optimisation procedure, where Eq. 6 is used repeatedly with different inputs and constraints. A numerical optimiser, employing a sequential quadratic programming method$^{41}$, is used throughout the algorithm to optimise the power, $\eta = \{\eta_1, ..., \eta_i\}$, and weighting, $r = \{r_1, ..., r_i\}$, variables for the $i$ input vectors. The algorithm first optimises the power using the Power optimisation method for one input vector. This power is employed for checking if adding any input vectors and numerically optimising only the weighting variables will increase the value of $\lambda_1(-(L+C))$. If the convergence speed is improved then the new selection of input vectors will have their power variables numerically optimised, before repeating the search for new input vectors and optimising the weighting variables. Once all input vectors have been checked both the weighting and power variables are optimised numerically. To check that there are not any redundant input

\begin{algorithm}
1: Use the $n$ ordered input vectors as defined in Algorithm 2.
2: Power optimisation of the first input vector to obtain an optimal value of $\eta$, see Eq. 5, by maximising $\lambda_1(-(L+C))$ where $C = \text{diag}(p_1)$.
3: Set $M = \lambda_1(-(L+C))$
4: for $i = 2$ to $n$
5: Include input vector $i$ with its associated power and weighting variables in the optimisation.
6: Set $q$ as the number of input vectors included in the optimisation, $r = \{1, ..., r_q\}$ and $\eta = \{\eta_1, ..., \eta_{q-1}\}$.
7: Optimise only the weighting variable $r_q$ for input vector $i$, using Eq. 6 to maximise $\lambda_1(-(L+C))$.
8: if $\lambda_1(-(L+C)) \times 1.001 > M$ then
9: Remove input vector $i$ with its associated power and weighting variables from the optimisation.
10: else if $\lambda_1(-(L+C)) \times 1.001 > M$ then
11: Set $M = \lambda_1(-(L+C))$, update the weighting variable $r_q$ with the optimised variable and set $\eta = \{\eta_1, ..., \eta_i\}$.
12: Optimise $\eta_1$ where $\eta = \{\eta_1, ..., \eta_i\}$, using Eq. 6 to maximise $\lambda_1(-(L+C))$.
13: Update the power variables with optimised values.
14: end if
15: end for
16: Optimise all weighting and power variables, using Eq. 6 to maximise $\lambda_1(-(L+C))$.
17: Set $M = \lambda_1(-(L+C))$ and $n$ as the number of input vectors still included in the optimisation.
18: for $i = 1$ to $n$
19: Calculate $C_i$ by removing input vector $i$ and its variables from Eq. 6 .
20: if $\lambda_1(-(L+C_i)) < M$ then
21: Remove input vector $i$ with its associated power and weighting variables from the optimisation.
22: Set $M = \lambda_1(-(L+C))$.
23: else if $\lambda_1(-(L+C_i)) > M$ then
24: Include input vector $i$ with its associated power and weighting variables in the optimisation.
25: end if
26: end for
27: Optimise all weighting and power variables, using Eq. 6 to maximise $\lambda_1(-(L+C))$.
\end{algorithm}
vectors in the optimisation each vector is removed from the optimisation, starting with the first input vector. If removing the
vector does not improve the performance it is reintroduced and the final combination of input vectors (i.e. communities) are
optimised numerically by varying their weighting and power variables to maximise $\lambda_1(-(L+C))$.

References

1. IDG. 2018 cloud computing survey. https://www.idg.com/tools-for-marketers/2018-cloud-computing-survey/ (2018).
   [Online; accessed June-2019].

2. Henry, C. Amazon planning 3,236-satellite constellation for internet connectivity. https://spacenews.com/
   amazon-planning-3236-satellite-constellation-for-internet-connectivity/ (2019). [Online; accessed June-2019].

3. Andrea, I., Chrysostomou, C. & Hadjichristofi, G. Internet of things: Security vulnerabilities and challenges. In 2015
   IEEE Symposium on Computers and Communication (ISCC), 180–187 (IEEE, 2015).

4. Braun, U., Muldoon, S. F. & Bassett, D. S. On human brain networks in health and disease. eLS (2015).

5. Roncal, W. G. et al. Migraine: Mri graph reliability analysis and inference for connectomics. In Global Conference on
   Signal and Information Processing (GlobalSIP), 2013 IEEE, 313–316 (IEEE, 2013).

6. Mišić, B., Goñi, J., Betzel, R. F., Sporns, O. & McIntosh, A. R. A network convergence zone in the hippocampus. PLoS
   computational biology 10, e1003982 (2014).

7. Bacik, K. A., Schaub, M. T., Beguerisse-Díaz, M., Billeh, Y. N. & Barahona, M. Flow-based network analysis of the
   caenorhabditis elegans connectome. PLoS computational biology 12, e1005055 (2016).

8. Malliaros, F. D. & Vazirgiannis, M. Clustering and community detection in directed networks: A survey. Phys. Reports
   533, 95–142 (2013).

9. Leicht, E. A. & Newman, M. E. Community structure in directed networks. Phys. review letters 100, 118703 (2008).

10. Page, L., Brin, S., Motwani, R. & Winograd, T. The pagerank citation ranking: Bringing order to the web. Tech. Rep.,
    Stanford InfoLab (1999).

11. Pons, P. & Latapy, M. Computing communities in large networks using random walks. In International symposium on
    computer and information sciences, 284–293 (Springer, 2005).

12. Shi, J. & Malik, J. Normalized cuts and image segmentation. Dep. Pap. (CIS) 107 (2000).

13. Bradley, P. S. & Fayyad, U. M. Refining initial points for k-means clustering. In ICML, vol. 98, 91–99 (Citeeseer, 1998).

14. Von Luxburg, U. A tutorial on spectral clustering. Stat. computing 17, 395–416 (2007).

15. Klemm, K., Serrano, M. Á., Eguíluz, V. M. & San Miguel, M. A measure of individual role in collective dynamics. Sci.
    reports 2, 292 (2012).

16. Bonacich, P. Some unique properties of eigenvector centrality. Soc. networks 29, 555–564 (2007).
17. Seidman, S. B. Network structure and minimum degree. *Soc. networks* **5**, 269–287 (1983).

18. Li, R.-H., Qin, L., Yu, J. X. & Mao, R. Finding influential communities in massive networks. *The Int. J. on Very Large Data Bases* **26**, 751–776 (2017).

19. Zhan, J., Guidibande, V. & Parsa, S. P. K. Identification of top-k influential communities in big networks. *J. Big Data* **3**, 16 (2016).

20. Li, J. *et al.* Most influential community search over large social networks. In *2017 IEEE 33rd International Conference on Data Engineering (ICDE)*, 871–882 (IEEE, 2017).

21. Stanoev, A., Smilkov, D. & Kocarev, L. Identifying communities by influence dynamics in social networks. *Phys. Rev. E* **84**, 046102 (2011).

22. Punzo, G., Young, G. F., Macdonald, M. & Leonard, N. E. Using network dynamical influence to drive consensus. *Sci. reports* **6**, 26318 (2016).

23. Fitch, K. & Leonard, N. E. Information centrality and optimal leader selection in noisy networks. In *Decision and Control (CDC), 2013 IEEE 52nd Annual Conference on*, 7510–7515 (IEEE, 2013).

24. Lin, F., Fardad, M. & Jovanović, M. R. Algorithms for leader selection in large dynamical networks: Noise-corrupted leaders. In *Decision and Control and European Control Conference (CDC-ECC), 2011 50th IEEE Conference on*, 2932–2937 (IEEE, 2011).

25. Patterson, S., McGlohon, N. & Dyagilev, K. Optimal k-leader selection for coherence and convergence rate in one-dimensional networks. *IEEE Transactions on Control. Netw. Syst.* **4**, 523–532 (2017).

26. Gan, Z., Shao, H., Xu, Y. & Li, D. Performance of leader-following consensus on multiplex networks. *Phys. A: Stat. Mech. its Appl.* **509**, 1174–1182 (2018).

27. Clark, R., Punzo, G. & Macdonald, M. Consensus speed optimisation with finite leadership perturbation in k-nearest neighbour networks. In *Decision and Control (CDC), 2016 IEEE 55th Conference on*, 879–884 (IEEE, 2016).

28. Young, G. F., Scardovi, L., Cavagna, A., Giardina, I. & Leonard, N. E. Starling flock networks manage uncertainty in consensus at low cost. *PLoS computational biology* **9**, e1002894 (2013).

29. Ballerini, M. *et al.* Interaction ruling animal collective behavior depends on topological rather than metric distance: Evidence from a field study. *Proc. national academy sciences* **105**, 1232–1237 (2008).

30. Landman, B. A. *et al.* Multi-parametric neuroimaging reproducibility: a 3-t resource study. *Neuroimage* **54**, 2854–2866 (2011).

31. Zuo, W., Zhang, D. & Wang, K. An assembled matrix distance metric for 2d pca-based image recognition. *Pattern Recognit. Lett.* **27**, 210–216 (2006).
32. Morey, R. A. et al. Scan–rescan reliability of subcortical brain volumes derived from automated segmentation. *Hum. brain mapping* **31**, 1751–1762 (2010).

33. MathWorks. Constrained nonlinear optimization algorithms. http://www.mathworks.se/help/optim/ug/constrained-nonlinear-optimization-algorithms.html (2015). [Online; accessed 2-March-2015].

34. Ng, A. Y., Jordan, M. I. & Weiss, Y. On spectral clustering: Analysis and an algorithm. In *Advances in neural information processing systems*, 849–856 (2002).

35. Cavagna, A. et al. Scale-free correlations in starling flocks. *Proc. Natl. Acad. Sci.* **107**, 11865–11870 (2010).

36. Emmerton, J. & Delius, J. D. 21 beyond sensation: Visual cognition in pigeons. *Vision, brain, behavior birds* 377 (1993).

37. Balister, P., Bollobás, B., Sarkar, A. & Walters, M. Connectivity of random k-nearest-neighbour graphs. *Adv. Appl. Probab.* **37**, 1–24 (2005).

38. Shao, H., Mesbahi, M. & Xi, Y. The relative tempo of discrete-time consensus networks. In *Control Conference (CCC), 2015 34th Chinese*, 7362–7367 (IEEE, 2015).

39. Jadabaie, A., Lin, J. & Morse, A. S. Coordination of groups of mobile autonomous agents using nearest neighbor rules. *Autom. Control. IEEE Transactions on* **48**, 988–1001 (2003).

40. Mesbahi, M. & Egerstedt, M. *Graph theoretic methods in multiagent networks* (Princeton University Press, 2010).

41. MathWorks. fminunc unconstrained minimization. http://uk.mathworks.com/help/optim/ug/fminunc-unconstrained-minimization.html (2015). [Online; accessed 2-March-2015].

**Acknowledgements**

This work was supported, in part, by the Engineering and Physical Sciences Research Council [EP/L505080/1].

**Contributions**

R.C., G.P. and M.M. devised the study; R.C. developed the algorithms, performed the analyses, wrote the paper and prepared the figures; All authors reviewed the manuscript.

**Corresponding author**

Correspondence to Ruaridh Clark.