Contents

Dynamical Community Detection in Directed Networks

Candidate number: 1048531

March 2021

1 Introduction

When extracting information from complex networks, community detection allows us to aggregate nodes into clusters according to various criteria. These criteria could include location, some node function or (GET MORE). Particularly interesting is the concept of flow-based community detection. Many networks are constructed from the collection of data relating to some dynamical process, e.g. itineraries on a transport network (REF), the spread of ideas in a citation network (REF AND GET MORE). Hence, the network's topology is highly influenced by the dynamical process which it represents. But it is also possible to turn this concept on its head when we consider that a network's structure may in turn influence a dynamical process which runs over it. It is possible for nodes and communities of nodes to be categorised and ranked based on how they will affect any such dynamical process. The paper *The many facets of community detection in complex networks* thus defines dynamical communities on a network as "blocks of nodes with different identities that trap the flow and channel it in different directions" [11]. An understanding of how topology will affect flow can then be put to many uses such as optimising the flow on a network in the cases of transport systems (REF WAN GUIMERO), vulnerability analysis by identifying key nodes vital to flow (REF DYNAMICAL IMPORTANCE PAPER) or predicting the movement of trade or passengers (REF ROSVALL, KALZA). For the majority of these flow-based methods the dynamic process being modelled is the probability wave
an ensemble of random walkers
of a random walker as it diffuses with time over a network. This useful abstraction allows network scientists to draw expertise from spectral graph theory, control theory and probability theory to extract all sorts of useful results.

A lot of the existing literature on spectral graph theory focuses particular attention on undirected networks, primarily due to the attractiveness of the symmetric adjacency matrix which allows for the derivation of appealing theoretical results and simplifications. For most systems, however, direction is nontrivial and the extension of measures derived for symmetric adjacency matrices is not always obvious (DISCUSS TRANSPORT etc.). Additionally, an exciting new branch of network science is looking at the effects of modelling a random walker on a network without the Markov property by constructing so-called ‘memory networks’ (REF SSL ROSVALL). These memory shown to produce networks have been show to prouced more accurate results in predictive modelling and less information loss (REFERENCE ROSVALL,SSL). However memory networks are directed even if the original network is not (REF) which further motivates our interest in studying popular network measures in the context of directed networks and asymmetric adjacency matrices.

Recent new research has begun to find applications and correlations for control theory’s model order reduction methods in network science [11]. The paper [10] proposes a control theory-based approach to quantifying similarities between nodes in a network with respect to some dynamical process running over them. With this ground-up approach it is possible to rederive popular network science community detection method such as Markov Stability and (GET THIS). The paper proposes a standard similarity measure defined by a weighted bilinear product of two vectors which can be augmemted to suit the need at hand. This project will use continuous and discrete-time similarity measures proposed in this paper to construct similarity matrices which can then be used for graph clustering.

The report will be organised as follows: first the relationship between control theory and random walks on a network will be established in Section (REF). Then the methods by which similarity matrices are constructed from this process will be discussed. Two discrete-time similarity measures and one continuous-time similarity measure are chosen for study in the remainder of the report and explained in greater detail. Though alternative methods exist (REF), spectral (clustering) will be used to optimise a quality function resulting from these methods and this will be briefly discussed in Section (REF). In Section (REF) the results of a applying these measures to a simple model network with clear dynamical substructures will be explored. In Section (REF) a similar method will be applied to an airline network extracted from a real world dataset (REF). In Section (REF) a different method of clustering, core-periphery detection, will be discussed and applied to this same airline network.

(MENTION ERGODICITY)

2 Flow on a network

When a graph is representative of some dynamical system, i.e. a network of roads, a social network, or in a more abstract sense, the flow of information through a citation network, it makes to consider how the the topological structure of a network might affect a process flowing over it. The standard way to study this is to consider injecting a random walker into the network at some time point $t = 0$. Analogous to the heat equation in dynamical systems theory, the probability wave for the random walker spreads diffusively throughout the graph, influenced only by structural characteristics such as link availability and weighting. In first-order networks, this random walker is assumed to have the Markov property, and its past trajectory will not affect its subsequent step. Dynamical community detection seeks to identify communities of
they would affect the trajectory of such ,..
nodes based on similarities with respect to how they affect the flow of such a walker. Nodes clusters may be compared based on their ability to 'trap' a walker for a longer period time, or perhaps based on how a walker will behave in the network after it passes through them.

3 Notation

Throughout this project we will use the following notation:

- v_i : a node on the network
- N : the number of nodes in the network
- $y \in \mathbb{R}^N$: column vector (FIX)
- $p \in \mathbb{R}^N$: row vector of probability densities for the (continuous or discrete-time) Markov process
- $\pi \in \mathbb{R}^N$: stationary probability (row) vector for the Markov process.

3.0.1 Random walks

When a walker is at a node v_i, the set of its transition (jump) probabilities out of that node form a vector. Considering an ensemble of walkers moving across the network in synchrony, the collection of these vectors forms a transition matrix for the Markov process T [4]. For a directed, weighted network the form of this is derived from the network structure where $T_{ij} = A_{ij}/s_i^{\text{out}}$. So A_{ij} is the weighting of the link from v_i

to v_j, and s_i^{out} is the total outward node strength of v_i. For a discrete time random walk (DTRW) the evolution of the process is described by the equation $p^{n+1} = p^n T$.

In discrete time, successive jumps of the walkers form a Markov chain which reaches stationarity once a probability distribution $p = \pi$ is reached such that further jumps do not change the probability distibribution for the walkers on the network, i.e.

$$\pi = \pi T. \tag{1}$$

Here, π and p are row vectors as is convention in diffusion processes [10].

(FIX:)The Perron-Frobenius theorem guarantees that the absolute value of the largest eigenvalue λ_1 of an asymmetric, primitive matrix (in this case $\lambda_1 = 1$) is unique and that the corresponding eigenvectors have all (NONNEGATIVE? - CHECK ALL PF STUFF) values. When the dynamics on a network are ergodic (EXPAND...). Thus the Perron-Frobenius theorem allows us to find the unique stationary probability distribution of a strongly connected network by simply calculating the left eigenvector associated with the network's dominant eigenvalue [9].

Because transition matrices are stochastic matrices and all rows sum to unity, we have $T1 = 1$ and hence the corresponding left eigenvector can be denoted π and represents the stationary distribution (1). Intuitively, once a system has reached equilibrium, further transitions T will not affect the probability distribution over the system.

Furthermore, the random walk Laplacian $L_{\text{rw}} = D^{-1}L = I - T$ can be derived and used in the master equation of a node-centric continuous-time random walk [4]

$$\dot{p} = -pL_{\text{rw}} = -p(I - T). \tag{2}$$

The node-centric random walk has independent identically disrtibuted inter-event times following a Poisson distribution Poi(λ) where λ sets the time-scale and can be normalised to unity without loss of generality [4]. It can be shown that walks defined by the node-centric CTRW are statistically the same as those defined by the DTRW. By setting $\dot{p} = 0$ it becomes clear that the formula for the stationary distribution is the same as that for the DTRW (1) [4].

The majority of directed (CHECK) theoretical networks satisfy these conditions of ergodicity [4] (REF) but this assumption often fails in real-world networks. Multiple techniques exist in the literature (REF ROSVALL,SSL) to allow us to make use of the Perron-Frobenius theorem in nonergodic systems and we will discuss solutions to this below in Section 4.

4 Synthesising ergodic dynamics

By the Perron-Frobenius theorem (see Appendix section **??**), a unique stationary distribution for the random walk processes described above only exists a process with for ergodic dynamics. Ergodic dynamics are only guaranteed if a network is composed of a single, strongly connected component [4]. Dynamics on directed, real-world networks rarely satisfy this criterion. Also, if a network contains nodes with no out-links (dangling nodes) then the transition matrix will be ill-defined.

In order to be able to make use of measures such as the stationary probability distribution and transition matrices, network scientists make small adjustments to the network in order to make dynamics ergodic. Popular methods include extracting a strongly connected component from the graph [9] and performing all analysis on that, or else augmenting the graph with a small about of teleportation [9, 4, 5]. Neither method is perfect; extracting a single connected component is only appropriate if the majority of nodes in a network already belong to a single strongly connected component so not much information is being lost. Introducing teleportation can water-down network structures, reduce overclustering and lead to resolution limits [10]. These effects can reduce the accuracy of popular methods such as Markov stability. Nonetheless, in the case of networks that are not mostly comprised of a single strongly connected component, it remains the best option.

A random walk with teleportation is defined on a network as follows: a walker now continues to move along paths defined by the transition matrix with probability α but also may teleport to any node in the network with probability $(1-\alpha)$. The only exception is that when a walker lands on a node with no out-links, it then must teleport with probability unity [4]. This preserves the T matrix's stochasticity so we are still guaranteed an eigenvalue and eigenvector of unity. Thus rate equation for the process becomes (CHECK)

$$p_i(t+1) = \alpha \sum_j p_j(t) T_{ji} + (1-\alpha) u_i, \qquad \alpha \in [0,1]. \tag{3}$$

Here u is some preference row vector. This leads to a new formal definition of the stationary probability density (PageRank) and the transition matrix (CHECK) [4]:

$$\pi = \alpha\pi T + (1-\alpha)u \tag{4}$$
$$\pi = (1-\alpha)u(I-\alpha T)^{-1} \tag{5}$$
$$(T_{\text{teleport}})_{\vec{i}} = \begin{cases} (1-\alpha)T_{\vec{i}} + \alpha u & \text{if } k_i > 0 \\ u \quad \text{if } k_i = 0 \end{cases} \tag{6}$$

Originally, the teleportation vector u was defined as a vector of uniform probabilites $(\frac{1}{N} \cdots \frac{1}{N})$ but more recent papers **??**, inspired by the observed correlation between a node's PageRank score and its in-strength have shown that defining u based on the in-strength of each vertix reduces the noise that is injected to the network by teleportation.

For this analysis, we opt for the teleportation option as described above. All measures calculated in this study will be based on the modified stationary probability distribution and transition matrix as described in Equations (4). This has the

5 Dynamical processes on a network

It is interesting to consider the movement of a random walker on a network in the context of linear systems theory as discussed in (REF). Starting by defining a system of very general linear processes, along with methods of quantifying their similarity, we can then extend this understanding to different aspects of continuous and discrete-time random walks [**find**]. Consider the system

$$\dot{y}(t) = \mathcal{A}y(t), \tag{7}$$

where the vector $y_i(t) \in \mathbb{R}^n$ represents the state of a process on the the network at node v_i and time t. For our case this will usually be a time-evolving vector of transition probabilities T_{ik} from v_i to all other nodes $\{v_k\}$. This generalised dynamical process has the solution $y(t) = y(0)\exp(\mathcal{A}t)$ and by specifying $\mathcal{A}$ we gain insight into the similarity between nodes with relation to various processes which run over the network structure.

Random walk processes can be related to control theory in the following way [10]. At time $t = 0$ a walker is placed on a node v_i, if the impulse response being observed in the system $p(t)$ is the probability of the walker being on each node at time t this will correspond to an impulse where $p_i(0) = 1$ and the rest of the system has

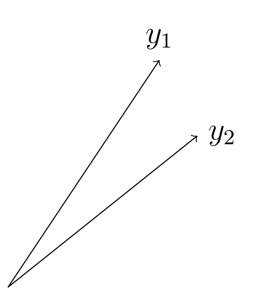
$p_j(0) = 0\, \forall j \neq i$. The impulse response of the system is then described by how the probabilities $p_j(t)$ of the system evolve over (discrete) time steps as the walker moves through the system.

We can assess the similarity of two vectors $\{y_i(t), y_j(t)\}$ by computing their bilinear product $\psi_{ij}(t) = \langle y_i(t), y_j(t)\rangle$ [10]. A weighting $\mathcal{W}$ can be added to this measure by computing $\psi_{ij}(t) = \langle y_i(t), y_j(t)\rangle_{\mathcal{W}} = y_i^\top \mathcal{W} y_j$. This weighting is often referred to as a null-model [10] and can be used to project away uninformative dimensions depending on what we are trying to learn. If the process is behaving similarly on nodes v_i and v_j at time t then it makes sense that this product will be large (CHECK? NORMALISED?). Gathering the column vectors $\{y_i(t)\}$ we form the matrix $n \times m$ matrix $Y(t) = \begin{pmatrix} y_1 & \dots & y_m \end{pmatrix}$ and thus the dynamical similarity matrix for a process $y(t)$is given by $\Psi(t) = Y^\top \mathcal{W} Y$. This similarity matrix also has a corresponding distance matrix defined by the Euclidean norm as [10]

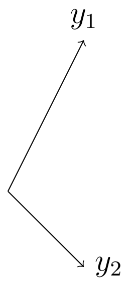
$$D_{ij}(t) = ||\mathcal{W}^{-\frac{1}{2}}(y_i(t) - y_j(t)||_2$$

This also generalises to the discrete case

$$y^{t+1} = M^\top y^t.$$



(a) y_1 and y_2 highly similar



(b) y_1 and y_2 not not very similar

Figure 1

For the purposes of this study we will focus on the propability vectors of the continuous and discrete time random walks (CTRW and DTRW). Here, we view y as probability column vectors in $\mathbb{R}^N$ where N is the number of nodes in the network. Let $y_0 = y(0)$, the inital value of y in both the continuous and discrete cases. In a directed, weighted network and for the discrete time random walk we have the set-up

$$y_{n+1} = T^\top y_n \qquad \text{where } (T)_{ij} = \frac{A_{ij}}{s_i^{\text{out}}} \tag{8}$$

$$y_n = [T^n]^\top y_0 \tag{9}$$

$$\Psi^n = T^n [T^n]^\top. \tag{10}$$

For the CTRW the set-up is:

$$\dot{y}(t) = -L_{\mathrm{rw}}^{\top} y \tag{11}$$

$$y(t) = e^{-L_{\mathrm{rw}}^{\top} t} y_0 \tag{12}$$

$$\Psi(t) = e^{-L_{\mathrm{rw}} t} e^{-L_{\mathrm{rw}}^{\top} t}. \tag{13}$$

These methods will be modified by the inclusion of various null models $\mathcal{W}$.

6 Specific similarity measures

The broad system description given above allows us to define time-dependent similarity matrices based on different properties of the flow. We will focus on three in particular,

6.0.1 Discrete-time unweighted similarity matrix

Considering a DTRW on the network defined by 5; we construct simple, unweighted similarity matrices $\Psi^n = T^n[T^n]^{\top}$ where n is timesteps. The i^{th} row of T^n represents the transition probability vector of the a walker originating at v_i after n steps $p_i^{(n)} = (T^n)_{\vec{i}}$. Thus, nodes v_i and v_j are said to be similar if their transition probability vectors after n steps are similar. This is equivalent to stating that two discrete-time random walkers tend to fall upon similar trajectories after leaving nodes v_i and v_j.

6.0.2 Walktrap similarity matrix

In the Walktrap algorithm, proposed in 2005 by Pascal Pons and Matthieu Latapy [], two nodes v_i and v_j are viewed as similar if DTRW walkers are likely to follow similar trajectories in n steps once they leave those nodes [4]. Again, this represented by the transition probability vectors p_i^n. The Walktrap measure also includes a null model which discounts the stationarity transition probabilities of random walker at step n (EXPLAIN BETTER). I.e. for all $k = 1, ..., N$, $p_i^{(n)}(k)$ represents the probability of the same walker transitioning from v_i to v_k in n steps and we discount the probability of a different independent walker just happening to move to v_k at step n. Hence, nodes are classed as similar in time n if walkers which originate from them have similar probability distributions over the network after n steps [**pons**, **lecs**]. In the original paper, the algorithm is defined for an undirected network with respect to the Euclidean distance between any two nodes transition probability vectors $T^n_{\vec{i}}$ and $T^n_{\vec{j}}$ as

$$r_{ij} = ||D^{-1/2}(T^n_{i\vec{\ell}} - T^n_{j\vec{\ell}})||_2 \tag{14}$$

Here, n should be chosen sufficiently large for a node to be able to traverse the entire network in n steps but not so large that both vectors $T^n_{\vec{i}}$ and $T^n_{\vec{j}}$ start approaching the stationary distribution of the network [4]. (DISCUSS HOW THIS WORKS IN NONERGODIC NETWORK) The weighting matrix for the undirected network is it's a diagonal matrix of its stationary distribution $D^{-1} = \text{diag}(\frac{1}{k_1} \dots \frac{1}{k_n}$ represents the probability of a walker visiting each node at equilibrium, regardless of it's origin. In the directed case this probability is given by the diagonal matrix of the stationary distribution $\Pi := \text{diag}(\pi)$. Where π and T are defined for nonergodic dynamics to include teleportation as described in (REFERENCE SECTION). As described in (REFERENCE SECTION), this translates to a similarity matrix

$$\Psi^n_\Pi = T^n \Pi [T^n]^\top. \tag{15}$$

6.0.3 Continuous-time dynamical similarity matrix

Dynamical similarity is defined in [10] (DO THIS) write this

6.1 Time-scales

6.2 Similarity-based clustering

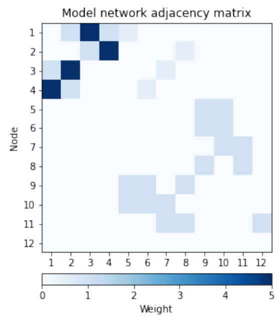
Once the similarity matrices have been constructed, they can be used to cluster the network into separate dynamical modules. This can be achieved by Louvain-like block detection or spectral clustering [10]. This project will use the `SpectralClustering()` function provided in Python's sklearn module [8] for this purpose. To give a brief overview of the methodology, spectral clustering methods operate as follows [4, 10, 7]: first the c eigenvectors corresponding the the dominant c eigenvalues of the similarity matrix $\Psi(t)$ are assembled, forming a new matrix $U \in \mathbb{R}^{N\times c}$. Letting the N vectors $\{y_i \in \mathbb{R}^c\}$, be the row vectors of this U, k-means clustering is then applied to assign each y_i to one of k clusters $\mathcal{C}_1, ..., \mathcal{C}_k$. (DIAGRAM WOULD BE NICE HERE) As is common practise, the `SpectralClustering()` function defaults to considering the same number of eigenvectors as clusters requested, $c = k$. This method has the disadvantage that it requires the user to specify the desired number of clusters in advance so cluster numbers will be entered manually, based on qualitative inspection of similarity matrices and what information is being considered.

7 Initial analysis on a model network

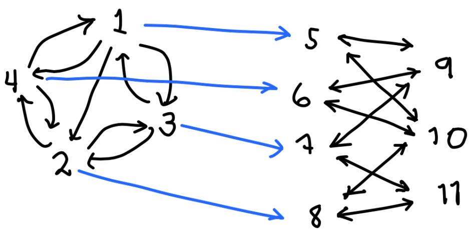
We will now perform a qualitative assessment of these similarity measures in the context of a model network that has an obvious dynamical structure. We look at a network originally presented in the paper [10]. (INSERT DIAGRAM INTO FIGURE 1) This network's setup and adjacency matrix is shown in Figure 2b. The network is directed, not strongly connected [10] and contains a bipartite substructure between the nodes $\{5, 6, 7, 8\}$ and $\{9, 10, 11\}$ while the nodes $\{1, 2, 3, 4\}$ have a cyclic substructure and walkers which leave this substructure cannot return. Thus the probability of a walker inhabiting that segment of the network may only decrease with time. Edge weightings are not listed in the paper so estimates are made based on the visuals provided. This network's multiple dynamical substructures make it a good model with which to highlight the effects that different similarity measures emphasise. In order to make visualisations clearer, all similarities in this section are scaled to $[0, 1]$ by left-multiplying each Y matrix in the bilinear product by

$$\mathrm{diag}\left(\frac{1}{||y_1||_2} \cdots \frac{1}{||y_N||_2}\right)$$

before calculating the inner products. Clusterings are selected using spectral clustering on the similarity matrices as discussed in Section 6.2. The results of performing clustering on these matrices at time $t = 20$ are shown at the bottom of this section in Figure 6.



(a)



(b)

Figure 2

7.0.1 Unweighted DTRW performance

The original paper looked at the unweighted similarity matrix 6.0.1 for a DTRW on the network using 5. The unweighted similarity matrix for this is shown in Figure 3 where the lighter sections indicate blocks of similar nodes. Initially, we visually inspect these matrices. At time $t = 2$ the network appears to have 4 main dynamical block $\{1, 2\}$, $\{3, 4\}$, $\{5, 6, 7, 8\}$ and $\{9, 10, 11\}$ with a substructure of $\{5, 6\}$ and $\{7, 8\}$ in $\{5, 6, 7, 8\}$. The random walk process settles into a more robust long-term state from timestep as low as $t = 6$. At time 8 four clear sub-structures are visible along the diagonal: $\{1,2\}$, $\{3,4\}$, $\{5,6,7,8\}$ and $\{9,10,11\}$. For longer time-scales the substructure in $\{1,2\}$ and $\{3,4\}$ disappears and the four nodes are grouped as one block.

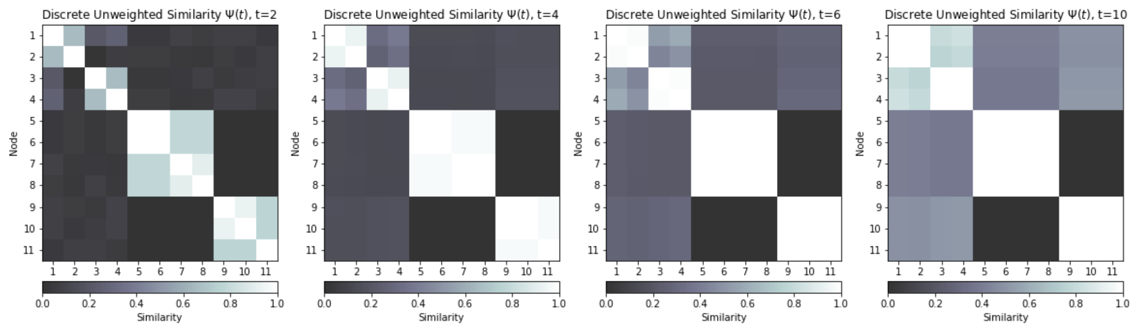


Figure 3: Unweighted DTRW similarity calculated from the bilinear product $T^n[T^n]^\top$ and normalised with respect to the product of the norms $||T_i||_2$, $||T_j||_2$ for visibility.

This simple similarity measure performs well on the model network and consistently isolates the 4 clusters $\{1,2\}$, $\{3,4\}$, $\{5,6,7,8\}$ and $\{9,10,11\}$ even for large time $t \sim 100$. When asked for a only two clusters, the algorithm isolates $\{5,6,7,8\}$ from the rest of the network. Nodes in the rest of the network are similar in that they have some probability of entering the $\{5,6,7,8\}$ group but nodes in $\{5,6,7,8\}$ can only travel between that group and $\{9,10,11\}$. When asked for three clusters the algorithm separates the remaining network into $\{1,2,3,4\}$ and $\{9,10,11\}$.

7.0.2 Walktrap

Next, the Walktrap similarity matrix Ψ_{Walktrap} is calculated for the network by the inclusion of a null model as described in section (REF SECTION). As the network is not ergodic, we use the PageRank, π (REF PAGERANK SECTION) instead of the stationary probability distribution and as usual, let $\mathcal{W} = \text{diag}(\pi)$. The similarity matrix generated is shown in Figure 4. As can be seen from the scale provided, the weighting matrix reduces the similarity values calculated as the similarity due to the stationary distribution is now being discounted (CHECK). In general, the blocks identified remain similar to those from the unweighted similarity matrix in the previous section.

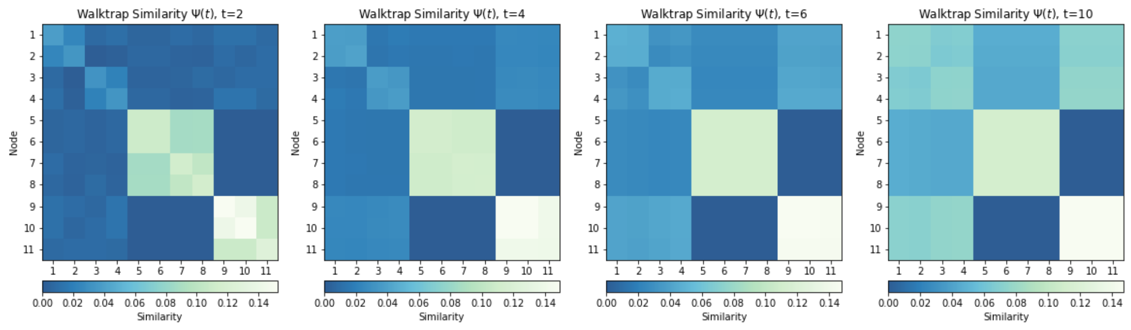


Figure 4: Walktrap similarity using the null model $\mathcal{W} = \Pi = \text{diag}(\pi)$ where π is the PageRank of the network. Calculated from the weighted bilinear product $\langle T^n, T^n \rangle_{\mathcal{W}}$ and normalised with respect to the product of the norms $||T_i||_2$, $||T_j||_2$ for visibility.

7.1 Dynamical Similarity

Finally, the dynamical similarity matrix $\Psi_{\text{dynamical}}$ is calculated as in 6.0.3 for times $\{2, 4, 6, 10, 20\}$. As can be seen in Figure 5, dynamical similarity isolates slightly different blocks to the previous two DTRW-based similarity matrices. Sub-clusters in the bi-parite network are visible for small time $t = 2$ but by $t = 5$ these have aggregated into one large sub-block representing the entire bi-partite network as one dynamical region. Spectral clustering is performed and the results for time $t = 20$ is shown in 6c. It does not appear to perform as well as the other two in this case in that it classifies node 6 separate from the rest of the network. The reason for this becomes apparent when reviewing the similarity matrices for this measure in Figure 5. For times greater than 10 the matrix has two distinct but homogeneous blocks. There are little-to-no differences within these blocks so the algorithm struggles to create further partitions for $k \geq 3$. This highlights the disadvangtages algorithms which require a user-specified number of partitions in advance. For smaller time scales (NOT SHOWN MAYBE GET PLOTS FOR APPENDIX) the algorithm groups $\{7, 8, 11\}$ together which is also in agreement with the similarity matrix in Figure 5.

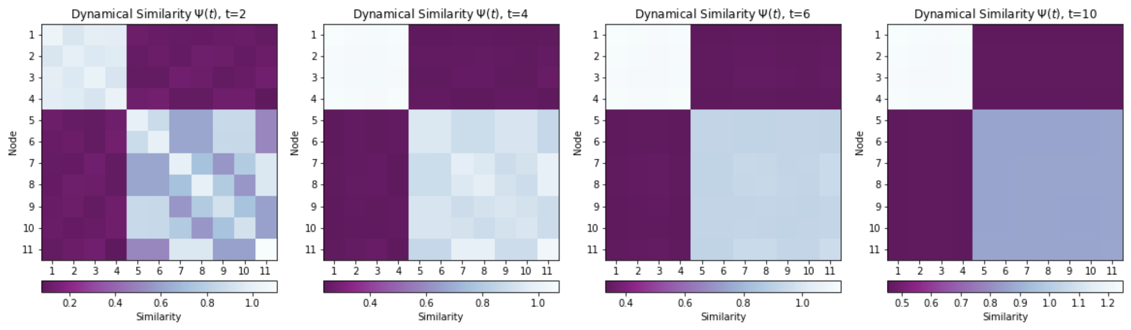
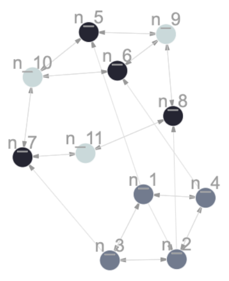
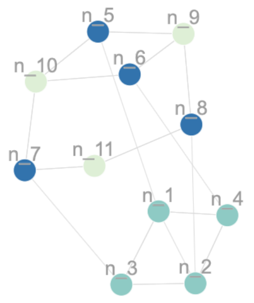


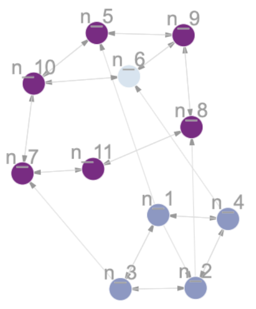
Figure 5: CTRW dynamical similarity matrix of the model network using the null model $\mathcal{W} = \Pi - \pi\pi^\top$ where π is the PageRank of the network. Calculated as described in Section 6.0.3 with each entry $(\Psi)_{ij}$ normalised with respect to the product of $||y_i||_2$ and $||y_j||_2$ for visibility.



(a)



(b)



(c)

Figure 6

8 Flight Network

We now consider the performance of these measures on a real world flight network. The first dataset is a set of passenger itineraries from the first quarter of 2011 and consists of flights to and from 103 Californian airports. The network contains 474 edges [1]. (LIT REVIEW OF FLIGHT NETWORKS). The network's adjacency matrix, shown Figure 7 is suggestive of some structure in the network. The presence of core-peripheral structures in airline networks is well-documented (REFERENCE AND EXPLAIN). The similarity measures for discrete dynamical, Walktrap and continuous dynamical similarity are all shown in Figures 8, 9 and 10. This time, the bilinear products are not normalised so the scale reduces with time for all plots (MAYBE FIX). Again, the two DTRW-based measures differ from the CTRW dynamical similarity. As hanging nodes have not been removed from the network prior to computing these, the black lines here represent the locations of such hanging nodes.The initial small-times visible here at time $t = 2$ are become less obvious as time increases and the nodes approach a more homogeneous stationary distribution.

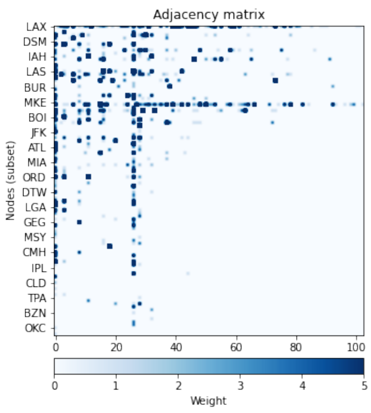


Figure 7: Heatmap of the weighted adjacency matrix for the flights network [1] with a subset of nodes listed on the y-axes.

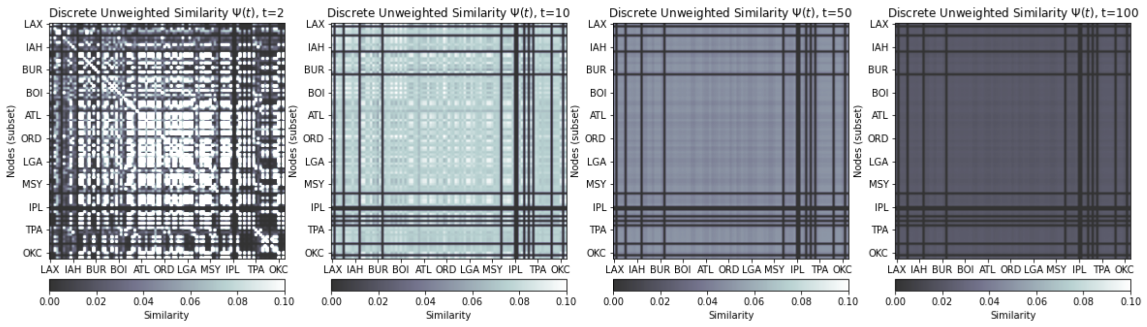


Figure 8: Unweighted DTRW similarity matrices for flights network.

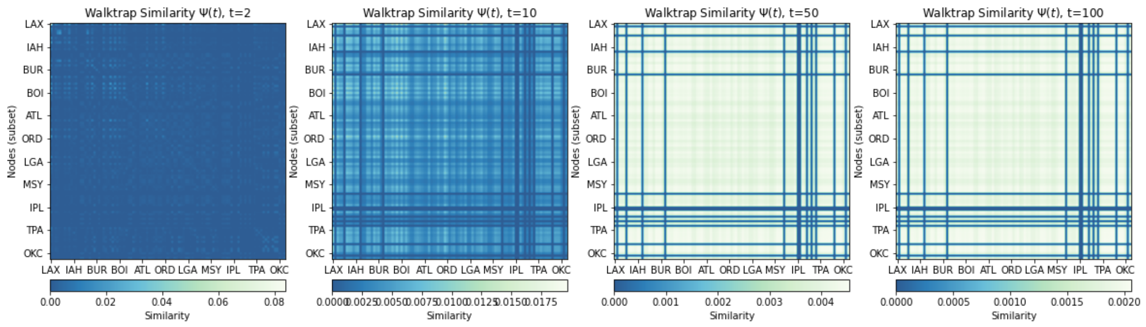


Figure 9: Walktrap similarity matrices calculated for the flights network.

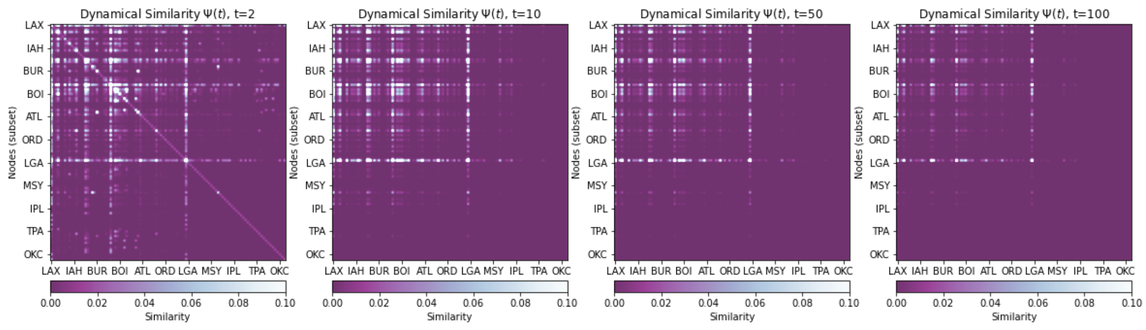
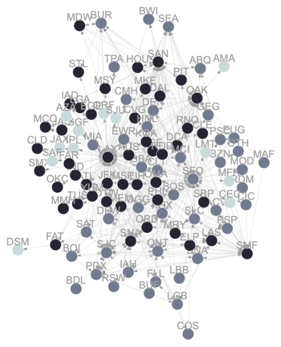
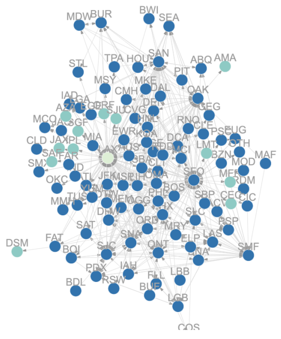


Figure 10: CTRW dynamical similarity matrices for the flights network.

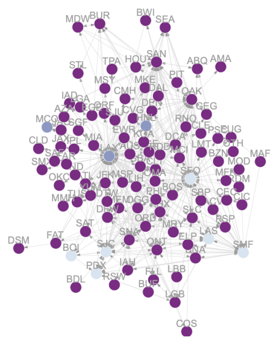
The results of spectral partitioning being performed on these matrices is visualised in Figures 11 and 12. (DISCUSS PREDICTIONS HERE).



(a) DTRW



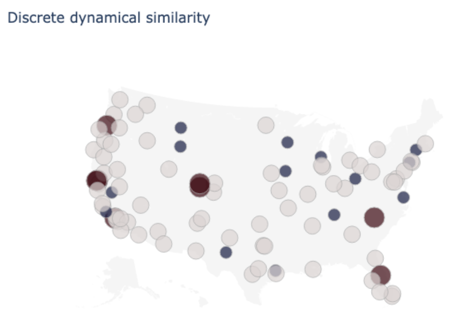
(b) Walktrap



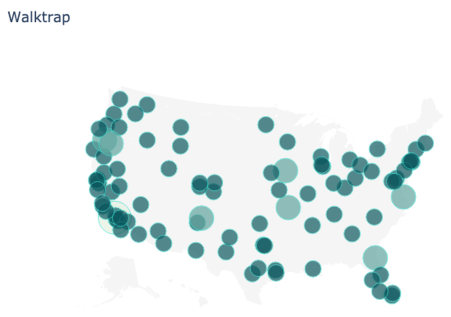
(c) CTRW

Figure 11

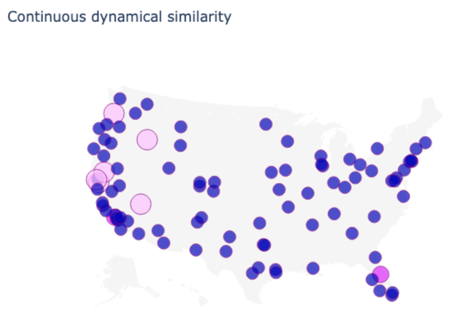
Flight predictions:



(a) DTRW



(b) Walktrap



(c) CTRW

Figure 12: Map visualisations

9 Dynamical Core-Periphery Structures

As mentioned above, the adjacency matrix of the flights network (Fig. 7) is indicative of what is known as a core-periphery structure [3] and indeed, core-periphery

are meso-scale structures and have been well-documented and researched in airline networks [3]. For comparison purposes we will perform core-periphery analysis on this network.
Many real-world networks display this type core-periphery structure. A network is partitioned into a group of core nodes, which are strongly connected and a set of peripheral nodes which are weakly connected to that core. The algorithm is discrete-time random walk-based and works under the understanding that if a walker is currently at a node that is part of the peripheral group, it's next step is very unlikely to also be in the peripheral group. A walker on a core node, however, is far more likely to remain within the well connected core or return to it quickly if it leaves [4]. To determine the core-periphery structure of a given network at stationarity we first consider our first walker at a node with the lowest out-strength and call this our initial peripheral set $\mathcal{S}_0$. There is an element of randomness in this algorithm but it has been shown to [not significant]. The persistence probability for a group $\mathcal{S}$ is defined as the probability that a walker will remain within that group on it's next step. For a group consisting of a single node this probability is naturally zero. The measure is defined as [4]

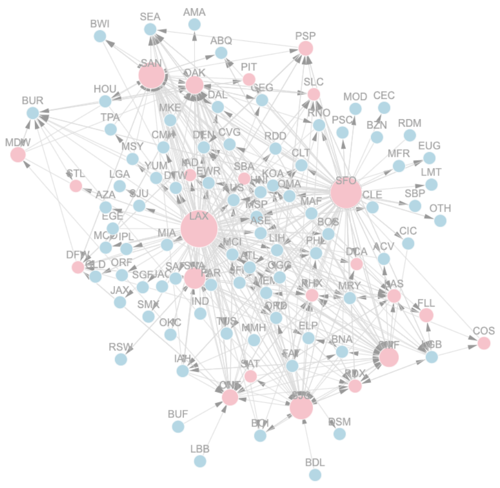
$$\alpha_{\mathcal{S}} = \frac{\sum_{i,j\in\mathcal{S}} \pi_i T_{ij}}{\sum_{i\in\mathcal{S}} \pi_i} = \frac{\text{Prob}\{\text{step to another node in } \mathcal{S}\}}{\text{Prob}\{\text{step to any other node}\}}$$

where π is the stationary probability density of the network and T is the transition matrix defined by $T_{ij} = A_{ij}/s_i^{\text{out}}$.
The algorithm is computationally expensive and works by successively adding the node that least increases the persistence probability to the group $\mathcal{S}_0$. As each node is added the amount by which the coreness of $\mathcal{S}$ increases is recorded and this values is assigned to the node as it's coreness value. Finally, a critical α-value is decided on and used to make a cut. Nodes with a high coreness value are assigned to the network's core and nodes with coreness below the threshold will be labelled as peripheral.

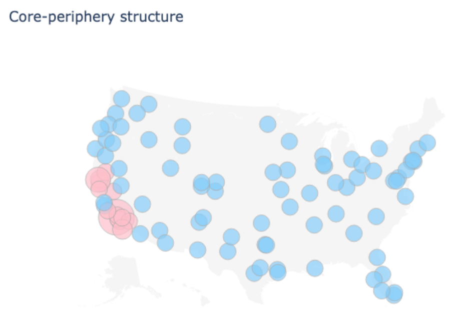
The algorithm works quite well on the airline network and identifies many of the airports you would expect to be hubs. In particular, it consistenly assigns LAX a far higher coreness score of 11.03 than the other nodes in the airport which agrees with the existing perception of LAX as a national and international hub airport. The airports with the top five coreness scores are shown in Table 1. Despite (REFERENCES SAYING STOCHASTICITY IRRELEVENT) the random choice element in the stocastic nature of the algorithm did lead to different groupings on different groupings. Las Angeles airport, for instance, was not always assigned a core-classification. Los Angeles is recognised as a major air traffic not only in the USA but internationally [2]. The tolerance for this section was set very low, $\alpha_c = 10^{-1}$, however, possibly lower than necessary. A slightly higher tolerance may reduce the stochastic effects of this algorithm. Other airports regularly classified as core airports include recognisable names such as Las Vegas, Burbank, Phoenix and Oakland. Visuals of the core-periphery grouping are included in Figure 13.

airport	coreness
LAX	7.8
SFO	3.534
SAN	1.406
SJC	0.971
SNA	0.686

Table 1: Airports with the highest five coreness scores.



(a)



(b)

Figure 13

10 Memory networks

Flow-based community detection and node-ranking measures are typically based on the probability paths of a random walker on the network. A walker is injected to a certain node on the network a $tt = 0$ and it's probabilities to be at different points of the network evolve with time. This random walker is assumed to have the Markov property, where it goes next depends on on it's current location and it immediately forgets where it was previous to that step. This assumption works well for diffusive dynamics, say, the spreading of get example but has been shown to be a significant oversimplification in many cases get ref such as, for example, in the flow of passengers in air travel where we expect a passengers next destination is highly conditional on where they came from. In these cases, the Markov assumption oversimplifies our understanding of dynamics on a network and valuable information is lost. Take about predictive flows.

11 Discussion

References

[1] *Air Carrier Statistics Database: T-100 Domestic Segment (U.S. Carriers)*. 2011.

[2] Roger Guimera et al. "The worldwide air transportation network: Anomalous centrality, community structure, and cities' global roles". In: *Proceedings of the National Academy of Sciences* 102.22 (2005), pp. 7794–7799.

[3] Sadamori Kojaku et al. "Multiscale core-periphery structure in a global liner shipping network". In: *Scientific reports* 9.1 (2019), pp. 1–15.

[4] Renaud Lambiotte. *C5.4 Networks Lecture Notes*. 2021.

[5] Renaud Lambiotte and Martin Rosvall. "Ranking and clustering of nodes in networks with smart teleportation". In: *Physical Review E* 85.5 (2012), p. 056107.

[6] Renaud Lambiotte, Martin Rosvall, and Ingo Scholtes. "From networks to optimal higher-order models of complex systems". In: *Nature physics* 15.4 (2019), pp. 313–320.

[7] Ulrike Von Luxburg. *A Tutorial on Spectral Clustering*. 2007.

[8] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[9] Vsevolod Salnikov, Michael T Schaub, and Renaud Lambiotte. "Using higher-order Markov models to reveal flow-based communities in networks". In: *Scientific reports* 6.1 (2016), pp. 1–13.

[10] Michael T Schaub et al. "Multiscale dynamical embeddings of complex networks". In: *Physical Review E* 99.6 (2019), p. 062308.

[11] Michael T Schaub et al. "The many facets of community detection in complex networks". In: *Applied network science* 2.1 (2017), p. 4.

A Theorems

(https://stanford.edu/class/ee363/lectures/pf.pdf)

Perron-Frobenius Theorem for Regular Matrices 1 *If A is a regular matrix, (i.e. A is non-negative and there exists $k > 0$ such that $(A^k)_{ij} > 0 \, \forall i, j$) then there exists an isolated eigenvalue λ_{pf} of A such that λ_{pf} is positive and $|\lambda| < \lambda_{pf}$ for all other eigenvalues λ. Also, the corresponding positive left and right eigenvectors are unique up to positive scaling [***boydnotes***].*

Perron-Frobenius Theorem for Nonnegative Matrices 1 *If A is a non-negative and there exists $k > 0$ such that $(A^k)_{ij} > 0 \, \forall i, j$ then there exists an eigenvalue λ_{pf} of A such that λ_{pf} is positive and $|\lambda| < \lambda_{pf}$ for all other eigenvalues λ. Also, the corresponding left and right eigenvectors are nonnegative (but are not necessarily unique or positive) [***boydnotes***].*

A.1 Overview of community detection methods

A.1.1 Markov stability

A.1.2 Walktrap

A.1.3 Core-Periphery methods

[6] [4]