COMMUNITY TREES IN NETWORKS

Yen-Chi Chen

Department of Statistics
University of Washington
Collaborators

Ruqian Chen  
UW Math

Wei Guo  
UW ISE

Ashis Banerjee  
UW ISE&ME
Outline

- Review: Density Tree
- Community Tree in Networks
- Future Work
Density Tree
Clusters and Density Function: an Illustration
Clusters and Density Function: an Illustration

level = 0.2
Clusters and Density Function: an Illustration

![Diagram of clusters with level = 0.2]
The idea of using a density level (threshold) $\lambda$ leads to clusters representing high density regions.

Thus, the level $\lambda$ has an effect on the clustering result.
The idea of using a density level (threshold) $\lambda$ leads to clusters representing high density regions.

Thus, the level $\lambda$ has an effect on the clustering result.

Varying the level $\lambda$ may lead to a creation of a new cluster or a merging of existing clusters.
Clusters and Density Function: Different Levels

level = 0.9
Clusters and Density Function: Different Levels

level = 0.7
Clustering and Density Function: Different Levels

level = 0.6
Clusters and Density Function: Different Levels

level = 0.5
Clusters and Density Function: Different Levels

level = 0.2
Clusters and Density Function: Different Levels

level = 0.1
- When the level changes, we see the *evolution* of clusters.
When the level changes, we see the *evolution* of clusters.

Cluster tree ([Stuetzle 2003](#)) is to summarize such an evolution process by a tree.
When the level changes, we see the *evolution* of clusters.

Cluster tree (*Stuetzle 2003*) is to summarize such an evolution process by a tree.

When applied to a density function, a cluster tree is also called a density tree (*Klemelä 2004*).
Density Tree: an Illustration
Density Tree: an Illustration
Density Tree: an Illustration
Density Tree: an Illustration
Density Tree: an Illustration
Density Tree: an Illustration
Density Tree: an Illustration
Density Tree: an Illustration
Density Tree: an Illustration
Density Tree: an Illustration
Density Tree: an Illustration
An Example of 2D Density Tree
Features of Density Trees

- Density trees provide topological information about the density function and they can be transformed into the persistent diagrams easily.

- When using density level sets to define clusters, the density tree contains the information about the evolution and stability of clusters.

- Moreover, density trees can always be displayed in two planes. So they are good tools for visualizing multivariate functions.
Features of Density Trees

- Density trees provide topological information about the density function and they can be transformed into the persistent diagrams easily.
- When using density level sets to define clusters, the density tree contains the information about the evolution and stability of clusters.
Features of Density Trees

- Density trees provide topological information about the density function and they can be transformed into the persistent diagrams easily.
- When using density level sets to define clusters, the density tree contains the information about the evolution and stability of clusters.
- Moreover, density trees can always be displayed in 2D plane. So they are good tools for visualizing multivariate functions.
Community Trees
We consider simple networks – undirected, unweighted graphs.

While there are many methods for analyzing a network, we focus on one particular method – the clique percolation method (CPM; Palla et al. 2005, 2007).
Community Detection in Networks

- We consider simple networks – undirected, unweighted graphs.
- While there are many methods for analyzing a network, we focus on one particular method – the clique percolation method (CPM; Palla et al. 2005, 2007).
- CPM is a popular and powerful method in community detection.
We consider simple networks – undirected, unweighted graphs.

While there are many methods for analyzing a network, we focus on one particular method – the clique percolation method (CPM; Palla et al. 2005, 2007).

CPM is a popular and powerful method in community detection.

CPM uses cliques and their overlapping to define communities.
Community Detection in Networks

- We consider simple networks – undirected, unweighted graphs.
- While there are many methods for analyzing a network, we focus on one particular method – the clique percolation method (CPM; Palla et al. 2005, 2007).
- CPM is a popular and powerful method in community detection.
- CPM uses cliques and their overlapping to define communities.
- Communities from CPM can be overlapping – this allows a broader way to interpret communities.
Finding level sets $\leftrightarrow$ CPM
Density level $\lambda$ $\leftrightarrow$ Clique order $k$
Clusters $\leftrightarrow$ Communities
Density/cluster trees $\leftrightarrow$ Community trees
Cliques and Communities

- Cliques: a subgraph such that all vertices are connected.
- $k$-clique: a clique with $k$ vertices.
Cliqués and Communities

- Cliqués: a subgraph such that all vertices are connected.
- \(k\)-clique: a clique with \(k\) vertices.
- Given a level \(k\), the CPM finds all \(k\)-cliques in the network and then forms an adjacency matrix \(A\) for these cliques.
- If the \(i\)-th and \(j\)-th \(k\)-cliques share the same \((k - 1)\) vertices, then \(A_{ij} = 1\) and 0 otherwise.
Cliques and Communities

- Cliques: a subgraph such that all vertices are connected.
- $k$-clique: a clique with $k$ vertices.
- Given a level $k$, the CPM finds all $k$-cliques in the network and then forms an adjacency matrix $A$ for these cliques.
- If the $i$-th and $j$-th $k$-cliques share the same $(k - 1)$ vertices, then $A_{ij} = 1$ and 0 otherwise.
- A $k$-clique community (or $k$-community for short) is a subgraph generated by the union of $k$-cliques in the same connected component of $A$. 
Cliquex and Communities

- Cliques: a subgraph such that all vertices are connected.
- $k$-clique: a clique with $k$ vertices.
- Given a level $k$, the CPM finds all $k$-cliques in the network and then forms an adjacency matrix $A$ for these cliques.
- If the $i$-th and $j$-th $k$-cliques share the same $(k - 1)$ vertices, then $A_{ij} = 1$ and 0 otherwise.
- A $k$-clique community (or $k$-community for short) is a subgraph generated by the union of $k$-cliques in the same connected component of $A$.
- The number $k$ of a clique community is called the order.
An Example: 4-communities

What are the 4-communities in the following network?
An Example: 4-communities

What are the 4-communities in the following network?
What are the 4-communities in the following network?
An Example: 4-communities

What are the 4-communities in the following network?
An Example: 4-communities

What are the 4-communities in the following network?
What are the 4-communities in the following network?
An Example: 4-communities

What are the 4-communities in the following network?
What are the 4-communities in the following network?
An Example: 4-communities

What are the 4-communities in the following network?
What are the 4-communities in the following network?

2nd 4-community
Another example of 4-communities (source: wikipedia).
What happens if we vary the number $k$?
What happens if we vary the number $k$?

A $k$-clique can be written as the union of $k$ distinct $(k - 1)$-cliques.

Thus, a $k$-community will be a subgraph of a $(k - 1)$-community.
Communities of Different Orders

- What happens if we vary the number $k$?
- A $k$-cliques can be written as the union of $k$ distinct $(k - 1)$-cliques.
- Thus, a $k$-community will be a subgraph of a $(k - 1)$-community.
- Then for a $k$-community $C_k$, there exists a sequence of subgraphs
  \[ C_k \subset C_{k-1} \subset \cdots \subset C_1, \]
  where $C_\omega$ is an $\omega$-community.
What happens if we vary the number $k$?

A $k$-clique can be written as the union of $k$ distinct $(k-1)$-cliques.

Thus, a $k$-community will be a subgraph of a $(k-1)$-community.

Then for a $k$-community $\mathcal{C}_k$, there exists a sequence of subgraphs

$$\mathcal{C}_k \subset \mathcal{C}_{k-1} \subset \cdots \subset \mathcal{C}_1,$$

where $\mathcal{C}_\omega$ is an $\omega$-community.

This property, which we refer to as the nested property, defines a tree structure of all (clique) communities within a graph.
Communities of Different Orders

- What happens if we vary the number $k$?
- A $k$-cliques can be written as the union of $k$ distinct $(k - 1)$-cliques. 
- Thus, a $k$-community will be a subgraph of a $(k - 1)$-community. 
- Then for a $k$-community $\mathcal{C}_k$, there exists a sequence of subgraphs

$$\mathcal{C}_k \subset \mathcal{C}_{k-1} \subset \cdots \subset \mathcal{C}_1,$$

where $\mathcal{C}_\omega$ is an $\omega$-community.

- This property, which we refer to as the nested property, defines a tree structure of all (clique) communities within a graph.
- The resulting tree is called the community tree.
Community Tree: an Example
A Key Insight (revisited)

Finding level sets ↔ CPM
Density level $\lambda$ ↔ Clique order $k$
Clusters ↔ Communities
Density/cluster trees ↔ Community trees
Community Tree

- Each node of the tree represents a clique community.
- This tree shows how each community evolves when we vary the order $k$. 
Community Tree

- Each node of the tree represents a clique community.
- This tree shows how each community evolves when we vary the order $k$.
- The merging node represents the order where multiple communities at a higher order are merged into the same community.
- In a sense, the community tree can be viewed as a generalization of the cluster tree to networks.
Community Tree

- Each node of the tree represents a clique community.
- This tree shows how each community evolves when we vary the order $k$.
- The merging node represents the order where multiple communities at a higher order are merged into the same community.
- In a sense, the community tree can be viewed as a generalization of the cluster tree to networks.
- Moreover, the community tree leads to a persistent diagram.
A component in a community tree is a series of nodes starting from a leaf that moves all the way down.
Components in a Community Tree

- A component in a community tree is a series of nodes starting from a leaf that moves all the way down.
- The birth time of a component is the highest order of its nodes.
- The death time of a component is the highest order that it merges into another component.
Components in a Community Tree

- A component in a community tree is a series of nodes starting from a leave that moves all the way down.
- The birth time of a component is the highest order of its nodes.
- Two components merge at an order if they share the same node. When two components merge, the one that has a lower birth time merged into the other component.
Components in a Community Tree

- A component in a community tree is a series of nodes starting from a leaf that moves all the way down.
- The birth time of a component is the highest order of its nodes.
- Two components merge at an order if they share the same node. When two components merge, the one that has a lower birth time merged into the other component.
- The death time of a component is the highest order that it merge into another component.
Using the birth and death time of components, we obtain the persistent diagram of a community tree.

Let \((b_1, d_1), \cdots, (b_K, d_K)\) be the birth time and death time of components of a community tree. The persistent diagram is

\[
\text{PD} = \{(d_i, b_i) : i = 1, \cdots, K\} \cup \{(d, b) : d = b\}.
\]
Example: Dophin Network

62 Dophins’ social network data with 159 edges.
Example: Zachary Karate Club Network

A social network data about Zachary karate club; 34 vertices and 78 edges.
For a network $G_0$, if we only perturb it a little bit, how will its community tree change?

Namely, we want to understand the stability of a community tree.
For a network $G_0$, if we only perturb it a little bit, how will its community tree change?

Namely, we want to understand the stability of a community tree.

However, quantifying the tree difference is not easy.
For a network $G_0$, if we only perturb it a little bit, how will its community tree change?

Namely, we want to understand the stability of a community tree.

However, quantifying the tree difference is not easy.

Here we measure their difference using the bottleneck distance between the corresponding persistent diagrams.
Given two persistent diagrams $\text{PD}_1$, $\text{PD}_2$, their bottleneck distance is

$$d_\infty(\text{PD}_1, \text{PD}_2) = \inf_{\gamma} \sup_{A \in \text{PD}_1} \|A - \gamma(A)\|_\infty,$$

where the infimum is taking over all bijective mappings between $\text{PD}_1$ and $\text{PD}_2$. 
Given two persistent diagrams $\text{PD}_1, \text{PD}_2$, their bottleneck distance is

$$d_\infty(\text{PD}_1, \text{PD}_2) = \inf_{\gamma} \sup_{A \in \text{PD}_1} \|A - \gamma(A)\|_\infty,$$

where the infimum is taking over all bijective mappings between $\text{PD}_1$ and $\text{PD}_2$.

Let $\text{PB}(T)$ be the persistent diagram of a community tree $T$. Then we define a distance $d_B$ for community trees $T_1$ and $T_2$ as

$$d_B(T_1, T_2) = d_\infty(\text{PB}(T_1), \text{PB}(T_2)).$$
○ Given two networks $G_1$ and $G_2$, let $T(G_1)$ and $T(G_2)$ be their corresponding community trees.

○ It turns out that the difference between their community trees are bounded by a quantity called the total star number $TSN(G_1, G_2)$:

**Theorem (Chen et al. 2017)**

Let $G_1$ and $G_2$ be two networks. Then

$$d_B(T(G_1), T(G_2)) \leq TSN(G_1, G_2).$$
Total Star Number

- The total star number

\[ TSN(G_1, G_2) = RSN(G_1, G_2) + RSN(G_2, G_1). \]

- RSN\((G_1, G_2)\) is the removal star number which is defined as

\[ RSN(G_1, G_2) = \min\{|V_0| : \nu(e) \cap V_0 \neq \emptyset \ \forall e \in E(G_1) \setminus E(G_2)\}, \]

where \(V_0\) is a collection of vertices and \(|V_0|\) is the number of elements in the set \(V_0\) and \(E(G)\) is the edge of a network \(G\) and \(\nu(e)\) is the vertices attached to the edge \(e\).
The total star number

\[ TSN(G_1, G_2) = RSN(G_1, G_2) + RSN(G_2, G_1). \]

\( RSN(G_1, G_2) \) is the removal star number which is defined as

\[ RSN(G_1, G_2) = \min \{ |V_0| : \nu(e) \cap V_0 \neq \emptyset \ \forall e \in E(G_1) \setminus E(G_2) \}, \]

where \( V_0 \) is a collection of vertices and \( |V_0| \) is the number of elements in the set \( V_0 \) and \( E(G) \) is the edge of a network \( G \) and \( \nu(e) \) is the vertices attached to the edge \( e \).

\( RSN(G_1, G_2) \) can be interpreted as the minimal number of vertices we need to remove so that \( G_1 \) is a subgraph of \( G_2 \).
Total Star Number

- The total star number

\[ \text{TSN}(G_1, G_2) = \text{RSN}(G_1, G_2) + \text{RSN}(G_2, G_1). \]

- \( \text{RSN}(G_1, G_2) \) is the removal star number which is defined as

\[ \text{RSN}(G_1, G_2) = \min\{|V_0| : \nu(e) \cap V_0 \neq \emptyset \ \forall e \in E(G_1) \setminus E(G_2)\}, \]

where \( V_0 \) is a collection of vertices and \( |V_0| \) is the number of elements in the set \( V_0 \) and \( E(G) \) is the edge of a network \( G \) and \( \nu(e) \) is the vertices attached to the edge \( e \).

- \( \text{RSN}(G_1, G_2) \) can be interpreted as the minimal number of vertices we need to remove so that \( G_1 \) is a subgraph of \( G_2 \).

- Informally, the total star number can be interpreted as the minimal number of vertices that the network difference can be attributed to.
Theorem (Chen et al. 2017)

Let $G_1$ and $G_2$ be two networks. Then

$$d_B(T(G_1), T(G_2)) \leq \text{TSN}(G_1, G_2).$$

- The TSN can be small while many edges are removed.
Theorem (Chen et al. 2017)

Let $G_1$ and $G_2$ be two networks. Then

$$d_B(T(G_1), T(G_2)) \leq TSN(G_1, G_2).$$

- The TSN can be small while many edges are removed.
- For instance, if $G_1$ is the same as $G_2$ except removing all edges connecting to a particular vertex of $G_1$, then $TSN(G_1, G_2) = 1$. 
Theorem (Chen et al. 2017)

Let $G_1$ and $G_2$ be two networks. Then

$$d_B(T(G_1), T(G_2)) \leq \text{TSN}(G_1, G_2).$$

- The TSN can be small while many edges are removed.
- For instance, if $G_1$ is the same as $G_2$ except removing all edges connecting to a particular vertex of $G_1$, then $\text{TSN}(G_1, G_2) = 1$.
- Computing the total star number does not require building a community tree.
Community Tree: an Example

Original Network

Community Tree

Persistence Diagram

Birth Time

Death Time

New Network - 1

Birth Time

Death Time

New Network - 2

Birth Time

Death Time

$d_\infty = 1$
Although the total star number provides a useful bound for community trees, it cannot be computed easily.

**Theorem (Chen et al. 2017)**

*Computing the total star number is an NP-complete problem.*

- Note that the proof relies only on one simple observation: computing the total star number is the same as finding the minimum vertex cover.
Future Work
Future Directions

- Practical algorithm for bounding the total star number.
- Visualization tool using community trees.
- Effects from stochastic updates on community trees.
- Connections to overlapping communities.
Thank You!

More details can be found in Chen et al. (2017): "A Note on Community Trees in Networks" (https://arxiv.org/abs/1710.03924)


