Lectures 1 – 5
Supplementary Material
and
Applications

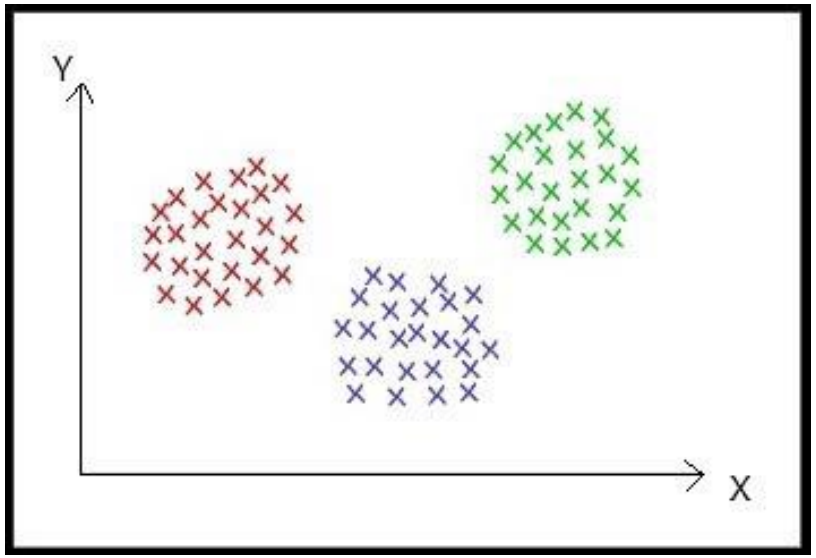
Community Detection Revisited

- *Community Detection* is equivalent to "*Clustering*" in unstructured data
- *Clustering*: Unsupervised machine learning
  - Find groups of nodes that are close to each other
  - Hundreds of methods published since 1950
  - Problem: what does "close to each other" means ?

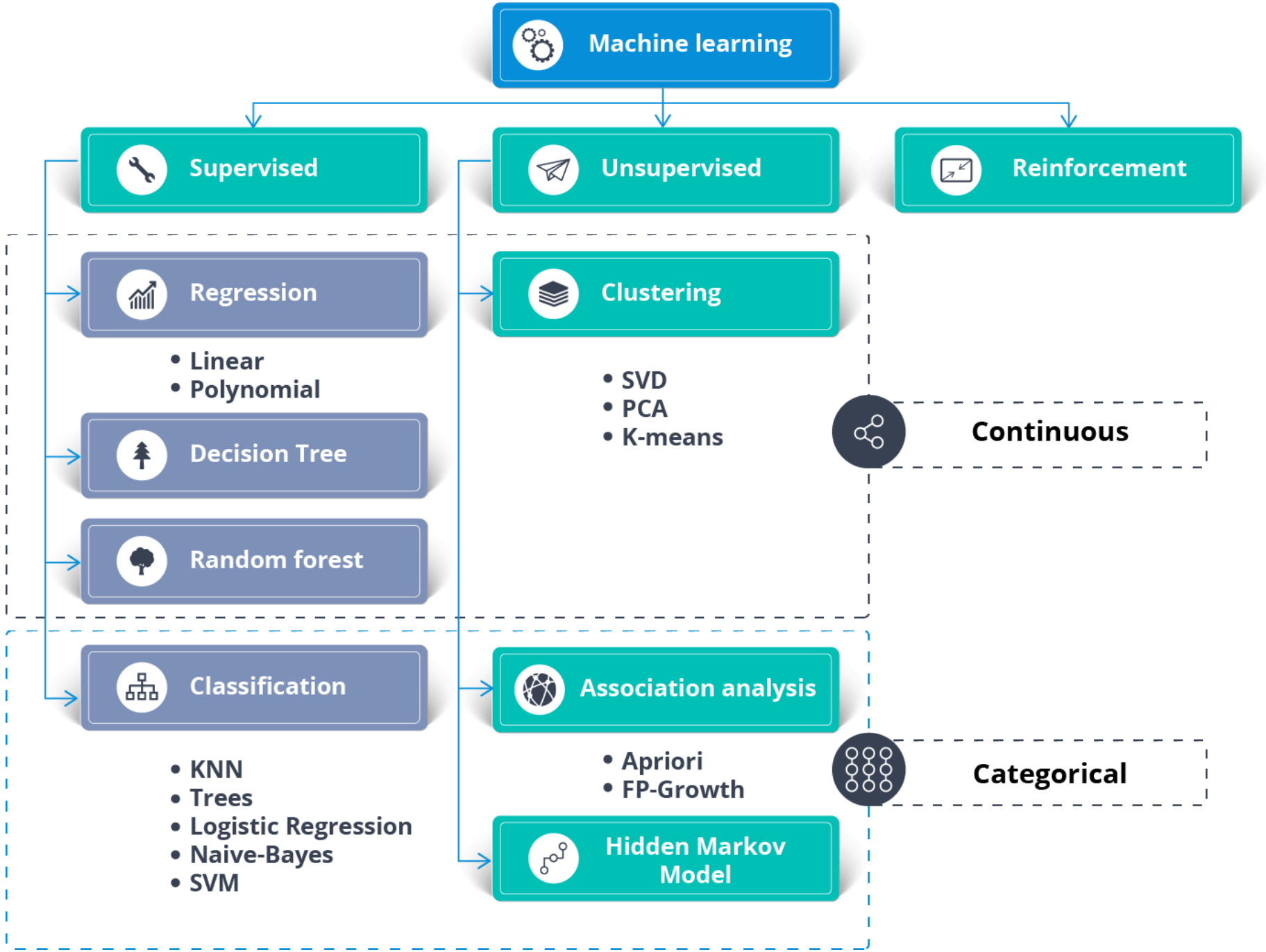
Clustering

Clustering is an *unsupervised* machine learning technique that divides the population into clusters such that data points in the same cluster are more similar and data points in different clusters are more dissimilar.

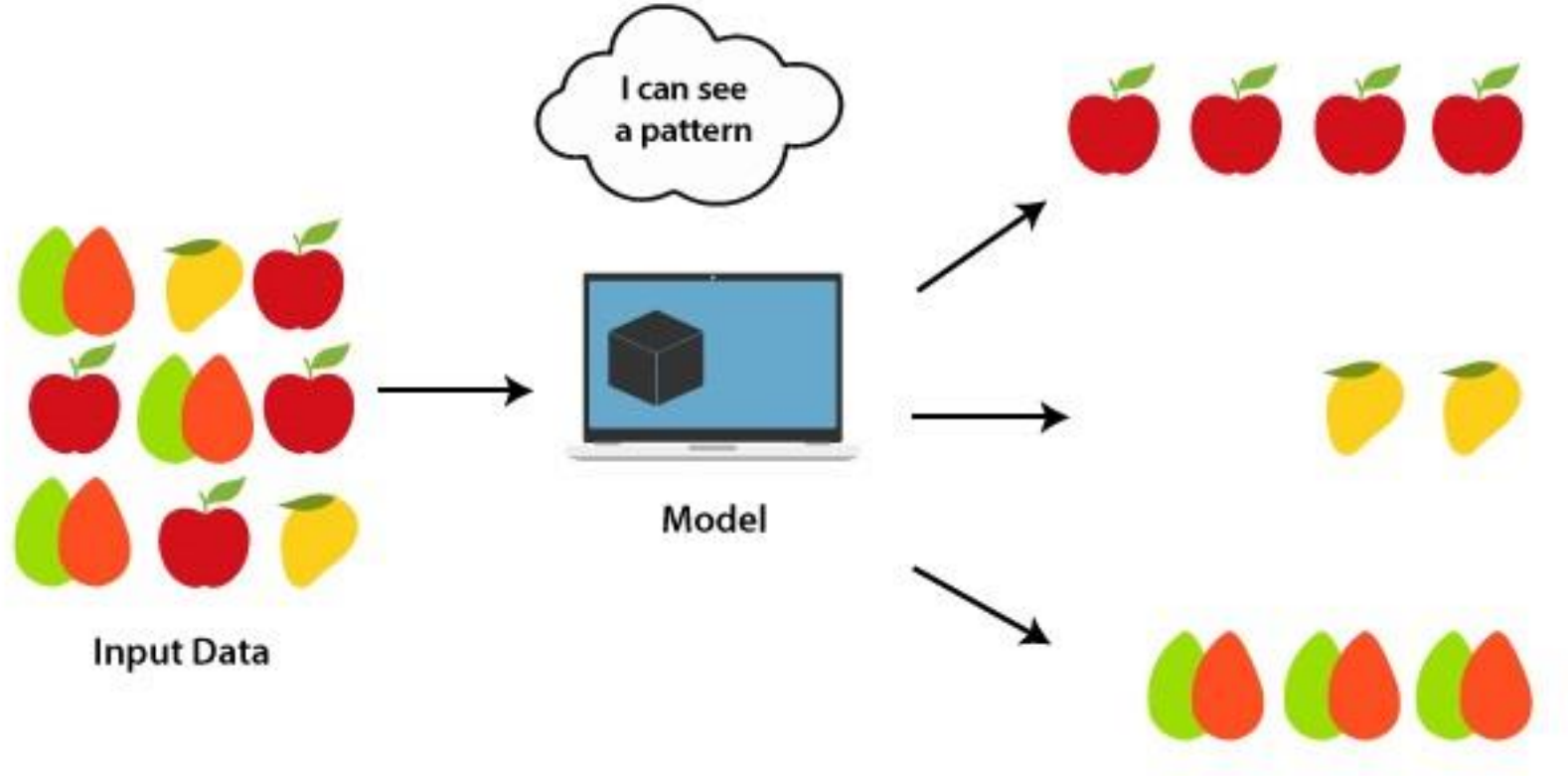
- Points in the same cluster are closer to each other.
- Points in the different clusters are far apart.



There are several types of clustering algorithms one can use, including ***K-Means Clustering***, ***DBSCAN***, *Hierarchical* ***Clustering*** *and many more ……*


Machine learning
Supervised
Unsupervised
Reinforcement
Regression
Linear
Polynomial
Decision Tree
Random forest
Clustering
SVD
PCA
K-means
Continuous
Classification
KNN
Trees
Logistic Regression
Naive-Bayes
SVM
Association analysis
Apriori
FP-Growth
Hidden Markov Model
Categorical

Unsupervised Learning



- Clustering
- Principal Component Analysis
- ……

Clustering Methods

- ***Clustering*** seeks to find *homogeneous subgroups* among the observations, so the members in each subgroup are close to each other.
- Find *homogeneity* and *heterogeneity* in the dataset.
- ***K-means clustering***: Partition observations into a *preset* number of clusters
- ***Hierarchical clustering***: No *preset* number of clusters, end up with a *tree-like* visual representation
- ……

Recall:

A good clustering method will produce high quality clusters with

- High intra-class similarity
- Low inter-class similarity

Quality of clustering depends on

- Similarity measure used by the method and its implementation
- Ability to discover some or all of the hidden patterns

Example: K-means Clustering

- ***K-means clustering*** is a simple and elegant approach for partitioning a dataset into ***K*** distinct, non-overlapping clusters
- To perform ***K-means clustering***, one needs to first specify the desired number of clusters ***K***
- The ***K-means algorithm*** assigns each observation to exactly one of the ***K*** clusters, by using a distance metric between two observations (e.g., Euclidean distance or correlation distance)

Example: K-means Clustering

- Let $C_1, \ldots, C_K$ be the ***K*** clusters of observations we wish to create. The set of clusters satisfies the following properties:

- $C_1 \cup C_2 \cup \ldots \cup C_K = \{1, 2, \ldots, n\}$.
  Each observation belongs to at least one of the ***K*** clusters.

- $C_k \cap C_{k'} = 0$ for all $k \neq k'$
  Each observation belongs to at most one of the ***K*** clusters.

→ The idea behind the ***K-Means Clustering*** is that a good clustering is to minimize the Within-Cluster Sum of Squares (WCSS) (i.e., variance).

Example: K-means Clustering

Within-Cluster Sum of Squares (WCSS)

The ***WCSS*** for cluster C_k, denoted by $W(C_k)$, is the amount by which the observations within a cluster differ from each other.

Using squared Euclidean distance,

$$W(C_k) = \sum_{x \in C_i} \|x - \mu_i\|^2 \ ,$$

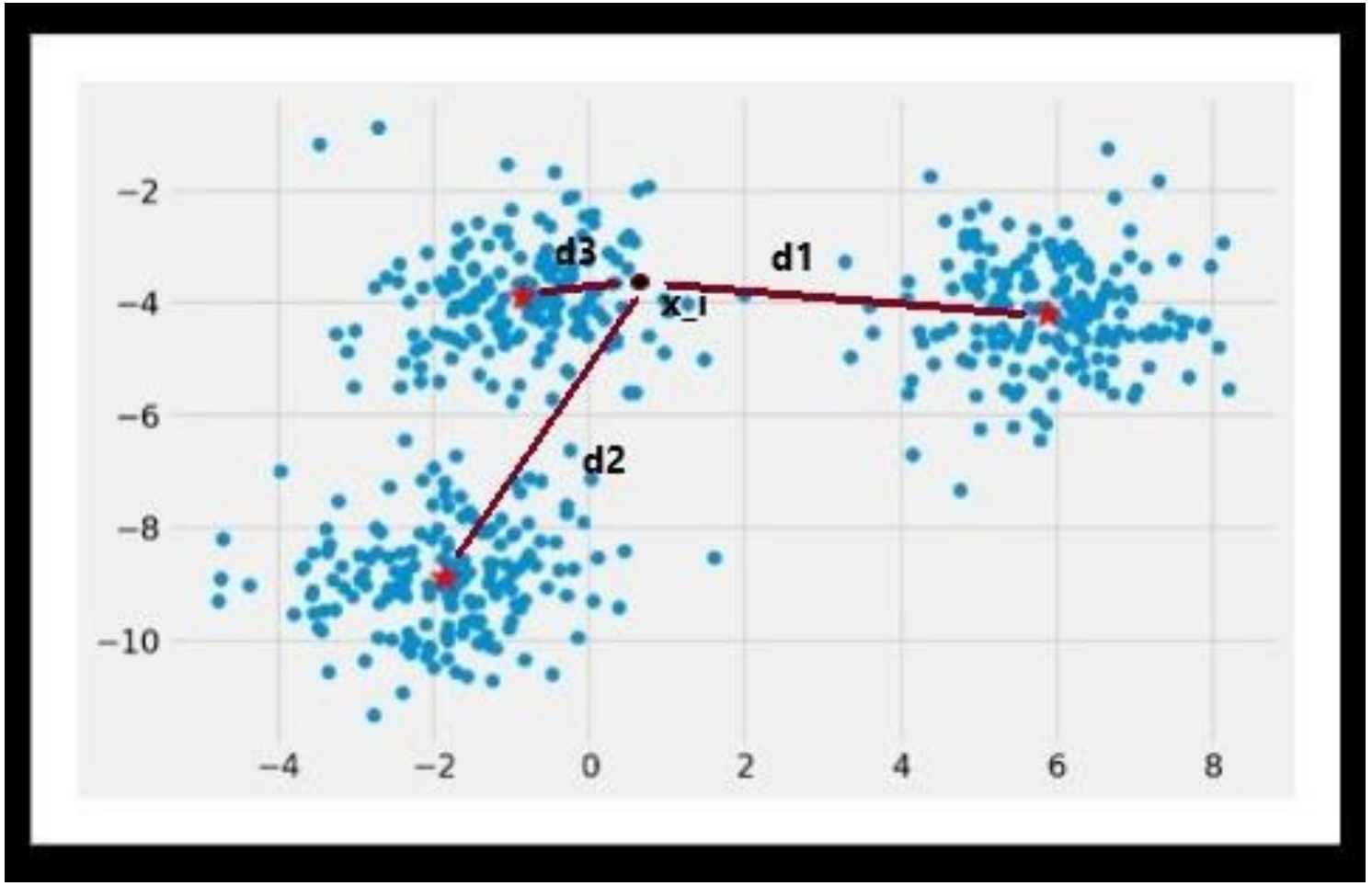
where μ_i denotes the centroid of the i^{th} cluster.

The best ***K-means clustering*** is the solution to

$$\min_{C_1, C_2, \ldots, C_K} \left\{ \sum_{k=1}^{K} W(C_k) \right\}.$$

*****This optimization problem is difficult to solve!!**

Example: K-means Clustering



The *K*-Means algorithm is to find *K*-centroid points and every point in the dataset will belong either of *K*-sets having minimum Euclidean distance.

K-means Algorithm

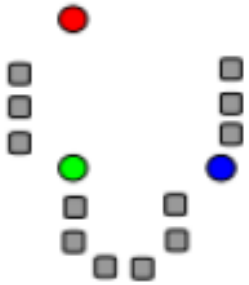
The most common algorithm uses an *iterative* refinement technique.

Given an initial set of ***K*** centroids, the algorithm proceeds by alternating between two steps:

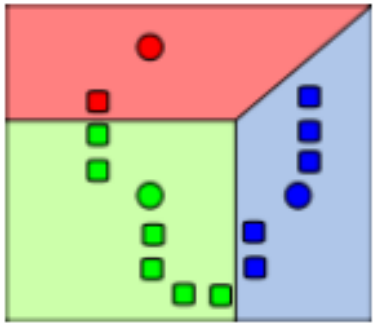
- ***Assignment step:*** Assign each observation to the cluster whose mean has the least squared Euclidean distance, this is intuitively the "nearest" mean.
- ***Update step:*** Calculate the new means to be the centroids of the observations in the new clusters.

Run the algorithm multiple times from different random initial configurations to obtain multiple local optima, then pick the best clustering result.

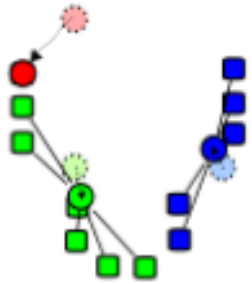
K-means Algorithm



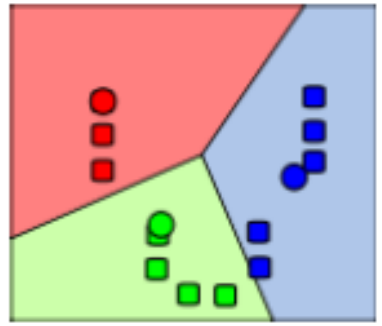
1. k initial "means" (in this case $k = 3$) are randomly generated within the data domain (shown in color).



2. k clusters are created by associating every observation with the nearest mean.



3. The centroid of each of the k clusters becomes the new mean.



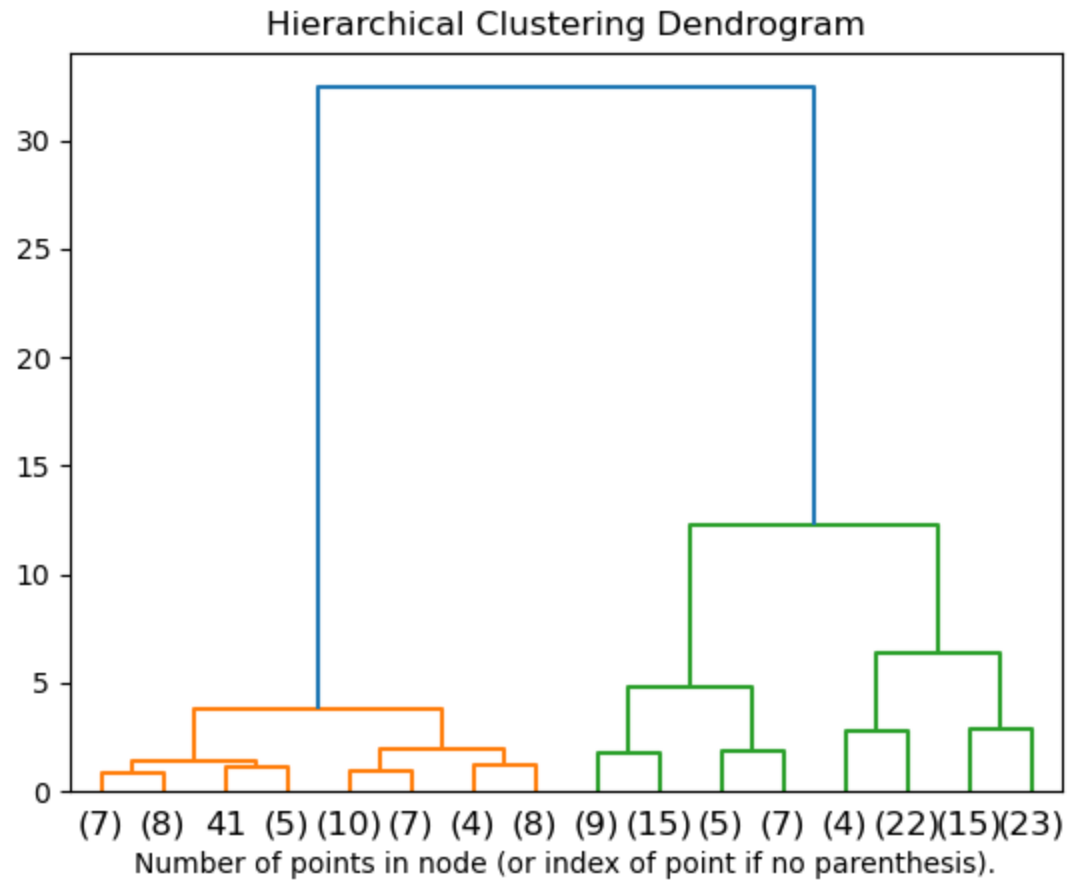
4. Steps 2 and 3 are repeated until convergence has been reached.

Hierarchical Clustering

Recall:

There are two types of hierarchical clustering methods:

- Divisive Clustering
- Agglomerative Clustering



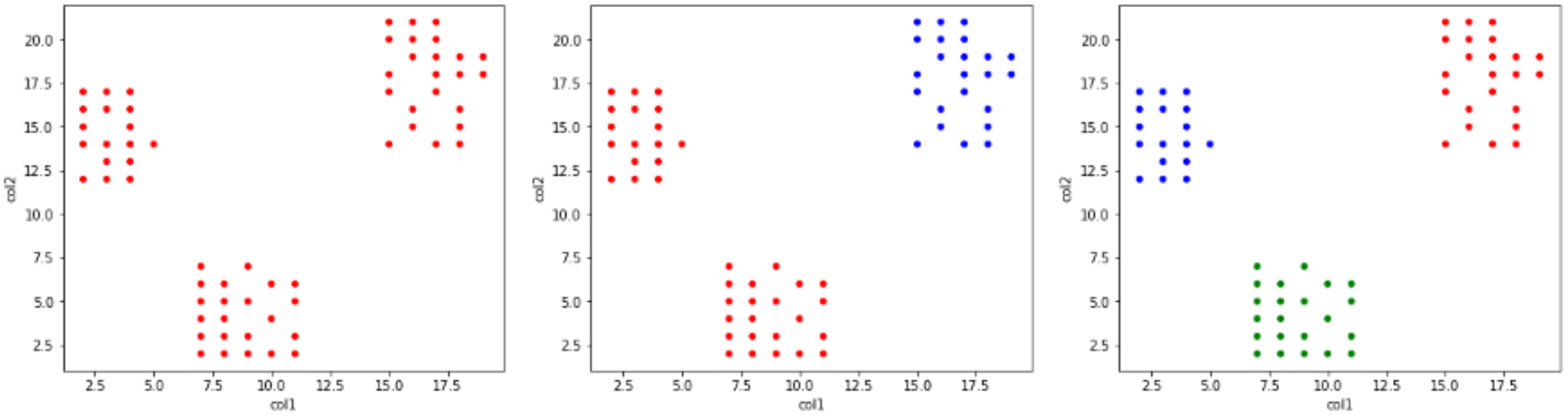
Divisive Clustering:

Divisive Clustering is a top-down clustering approach, initially, all the points in the dataset belong to one cluster and split is performed recursively as one moves down the hierarchy. Divisive clustering with an exhaustive search is **O**(2^n), but it is common to use faster heuristics to choose splits, such as ***K***-means.

Steps of Divisive Clustering:

1) Initially, all points in the dataset belong to one single cluster.
2) Partition the cluster into two least similar clusters
3) Proceed recursively to form new clusters until the desired number of clusters is obtained.

Divisive Clustering:



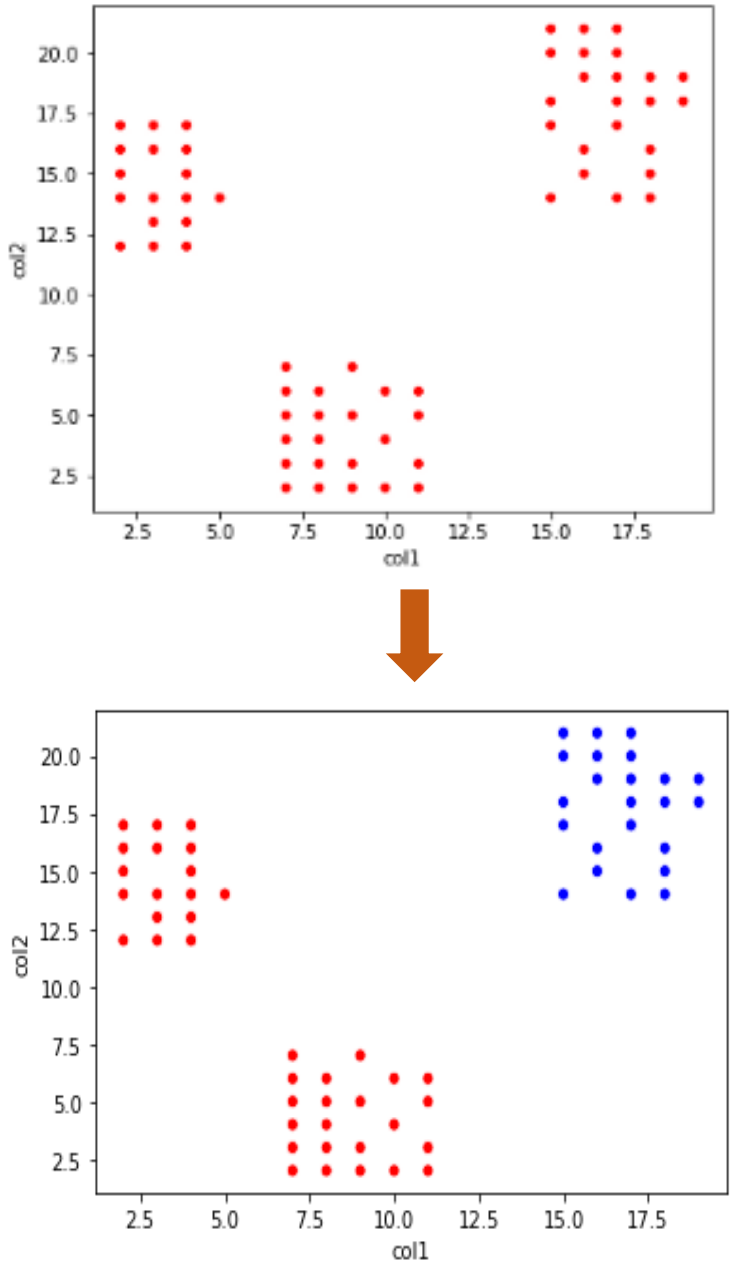
Left: All the data points belong to a single cluster; Middle: One cluster is separated from the previous single cluster; Right: Another cluster is separated from the previous set of clusters.

In the above dataset, the 3 clusters are separated from each other. Therefore, we stop here.

Divisive Clustering:

How to choose which cluster to split?

Check the Sum of Squared Errors (*SSE*) of each cluster and choose the one with the largest value. In the 2-dimension dataset shown on the right, the data points are separated into 2 clusters, for further separating the dataset to form the 3rd cluster, find *SSE* for each of the points in a red cluster and blue cluster.



Divisive Clustering:

How to split the chosen cluster?

After you decide to split which cluster, the question will be how to split the chosen cluster into 2 clusters. One way, e.g., is to use Ward's criterion to get the largest reduction in the difference in the *SSE* criterion as a result of the split.

How to handle the noise or outlier?

The presence of outlier or noise can result in forming a new cluster of its own. One way to handle the noise in the dataset is to use a threshold to determine the termination criterion which does not generate clusters that are too small.

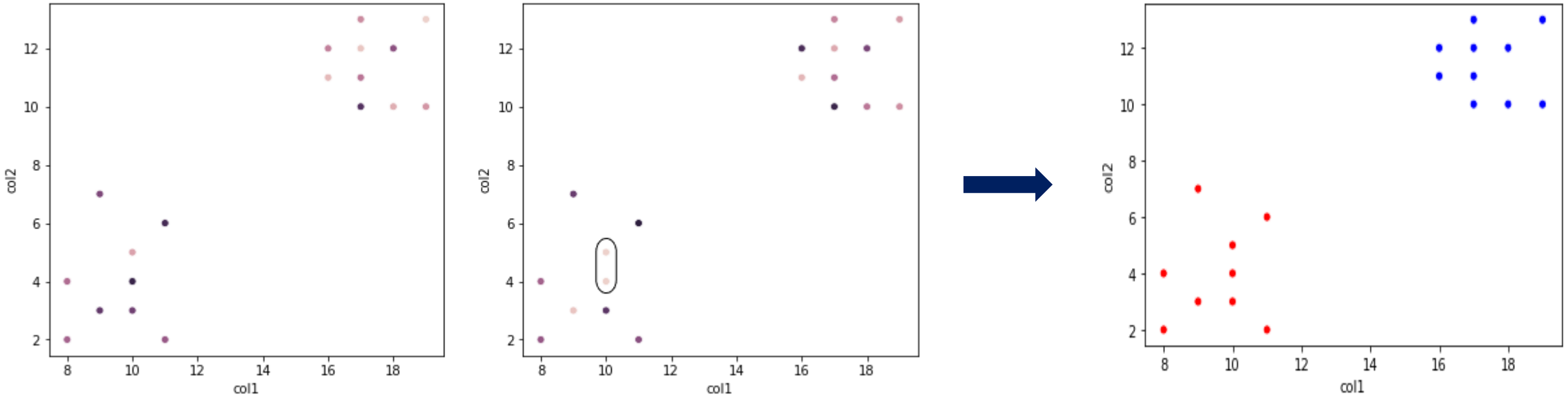
Agglomerative Clustering:

Agglomerative Clustering is a bottom-up approach, initially, each data point is itself a cluster of its own, further pairs of clusters are merged as one moves up the hierarchy.

Steps of Agglomerative Clustering:

1) Initially, all the data-points are a cluster of its own.
2) Take two nearest clusters and join them to form one single cluster.
3) Proceed recursively step 2 until one obtains the desired number of clusters.

Agglomerative Clustering:



Left: Every data point is a cluster of its own.
Right: Two nearest clusters (surrounded by a black oval) join together to form a new single cluster.

In the dataset above, 2 clusters are far separated from each other. So we stop after getting 2 clusters.

Agglomerative Clustering:

How to join two clusters to form one cluster?

To obtain the desired number of clusters, the number of clusters needs to be reduced from the initial *N* cluster (*N* is the total number of data points). Two clusters are combined by computing the similarity between them.

There are methods which are used to calculate the similarity between two clusters:

- Distance between two closest points in two clusters.
- Distance between two farthest points in two clusters.
- The average distance between all points in the two clusters.
- Distance between centroids of two clusters.

Agglomerative Clustering:

The standard algorithm for *Hierarchical Agglomerative Clustering* (*HAC*) has a time complexity of $\mathbf{O}(n^3)$ and requires $\mathbf{\Omega}(n^2)$ memory. For some special cases, optimal efficient agglomerative methods (of complexity $\mathbf{O}(2^n)$) are known: ***SLINK*** for single-linkage and ***CLINK*** for complete-linkage clustering. With a *heap* (specialized tree-based data structure), the runtime of the general case can be reduced to $\mathbf{O}(n^2 \log n)$, an improvement from $\mathbf{O}(n^3)$, at the cost of further increasing the memory requirements. In many cases, the memory overheads of this approach are too large to make it practically usable.

Except for the special case of single-linkage, none of the algorithms (except exhaustive search in $\mathbf{O}(2^n)$) can be guaranteed to find the optimum solution.

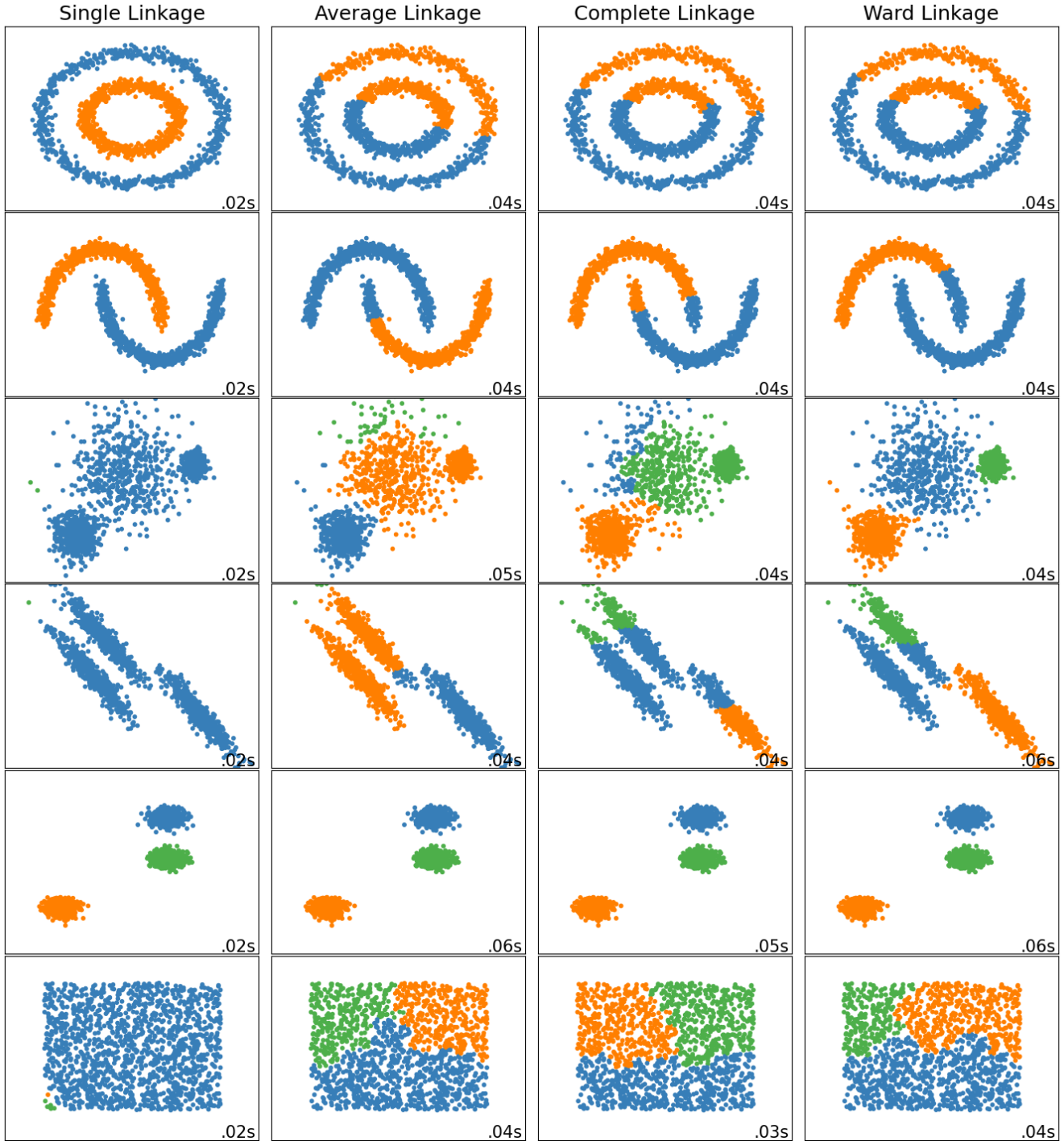
Agglomerative Clustering:

AgglomerativeClustering (scikit-learn) supports *Ward*, *single*, *average*, and *complete* linkage strategies.

- ***Ward*** minimizes the sum of squared differences within all clusters. It is a variance-minimizing approach and in this sense is similar to the k-means objective function but tackled with an agglomerative hierarchical approach.
- ***Maximum*** or ***complete linkage*** minimizes the maximum distance between observations of pairs of clusters.
- ***Average linkage*** minimizes the average of the distances between all observations of pairs of clusters.
- ***Single linkage*** minimizes the distance between the closest observations of pairs of clusters.

Agglomerative cluster has a "rich get richer" behavior that leads to uneven cluster sizes. In this regard, *single* linkage is the worst strategy, and *Ward* gives the most regular sizes. However, the affinity (or distance used in clustering) cannot be varied with *Ward*, thus for non Euclidean metrics, *average* linkage is a good alternative. *Single* linkage, while not robust to noisy data, can be computed very efficiently and can therefore be useful to provide hierarchical clustering of larger datasets. *Single* linkage can also perform well on non-globular data.

Different linkage type: Ward, complete, average, and single linkage

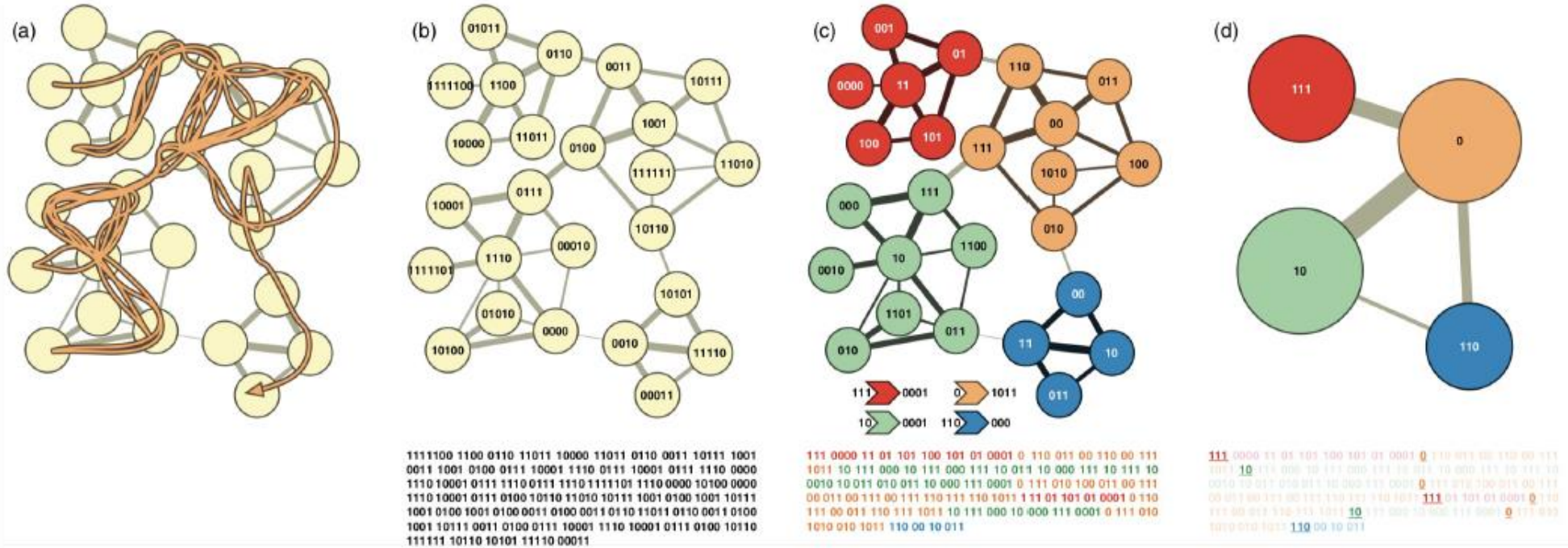


Community Detection: Summary (Recall)

- The Optimal Method?
  - It varies depending on applications, networks, computational resources etc.

- Other lines of research
  - Communities in directed networks
  - Overlapping communities
  - Community evolution
  - Group profiling and interpretation

Communities in Directed/Weighted Networks ?



For example: ***Infomap*** (using information flow)

Overlapping Communities

- In real networks, communities are often overlapping
  - *Some of your High-School friends might also be your University friends*
  - *A colleague might be a member of your family*
  - …
- Overlapping community detection is considered ***much harder***
  - *Not well defined*
  - *Nevertheless, many algorithms: K-Clique Percolation, etc …..*
- Border between “attributes” and overlapping communities ?
  - *Community of Women, Community of 17-19 yo, Community of fans of …*

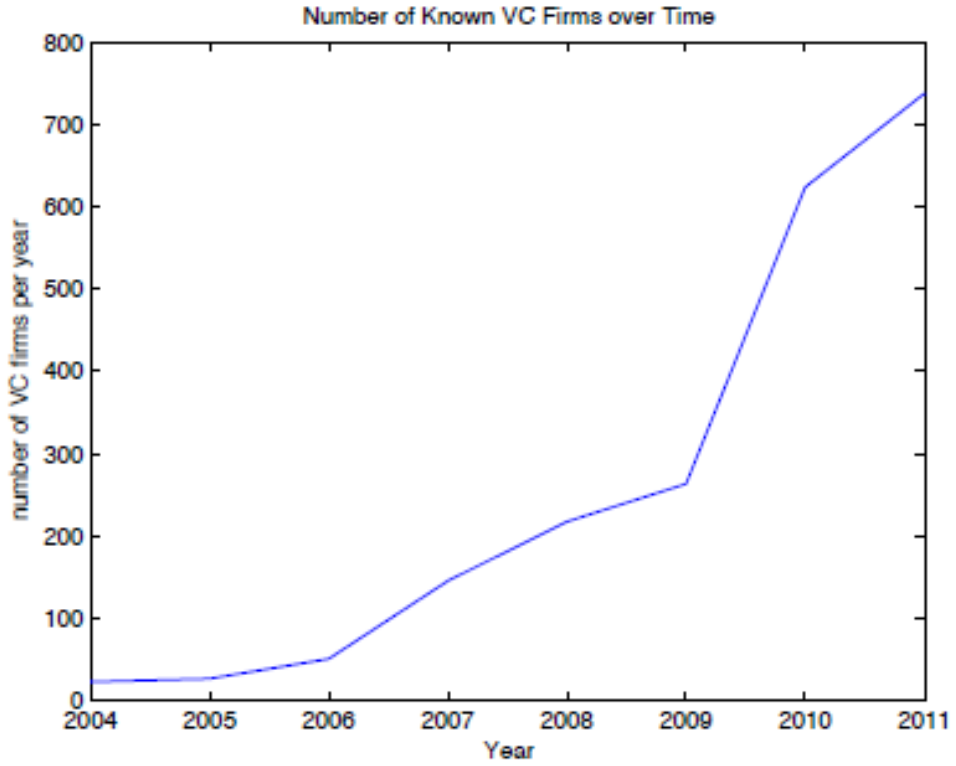
Case Study I:
Chinese Venture Capital Network

The Chinese Venture Capital Industry

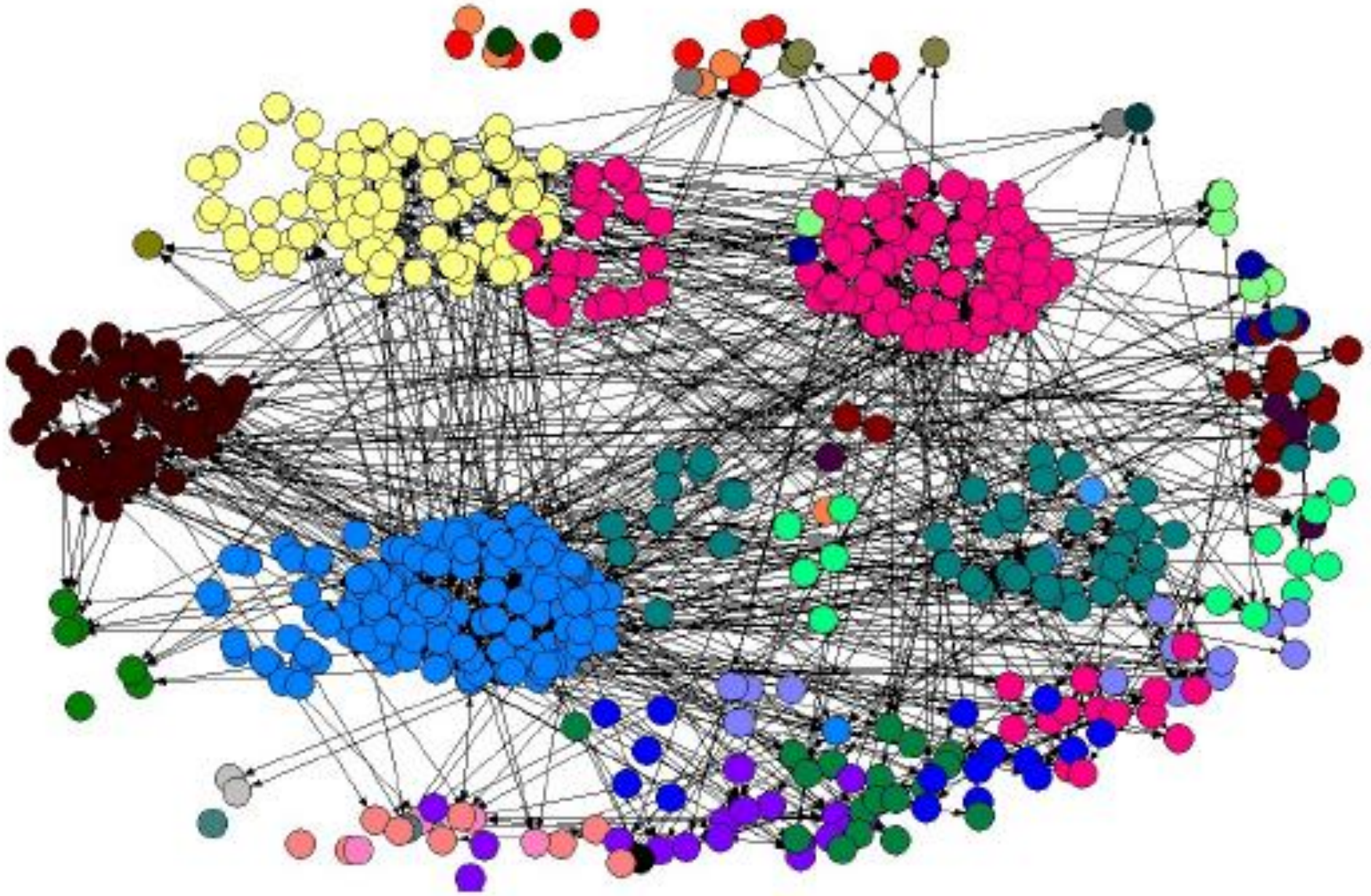
Since 2011, China has become the second largest economy in the world.

Research showed that small and medium enterprise owners in China relied mainly on the financial support from their own savings and immediate family in the past due to the stringent listing requirements.

Chinese government attempted to promote venture capital to fill the SME finance gap from the 1980s, and the Chinese VC industry has grown significantly since the late 1990s.



Number of Chinese venture capital companies from 2004 to 2011.



The VC Network of China. Nodes of the same color belong to the same province or district.

Statistical and Topological Properties of the VC Network

(1) Mean Shortest Path

$$\bar{l} = \frac{2}{N(N-1)} \sum_{i \geq j} d_{ij}$$

where d_{ij} is the distance of the shortest path between nodes *i* and *j*.

(2) Degree

The degree of a node is the number of links that are associated with this node.

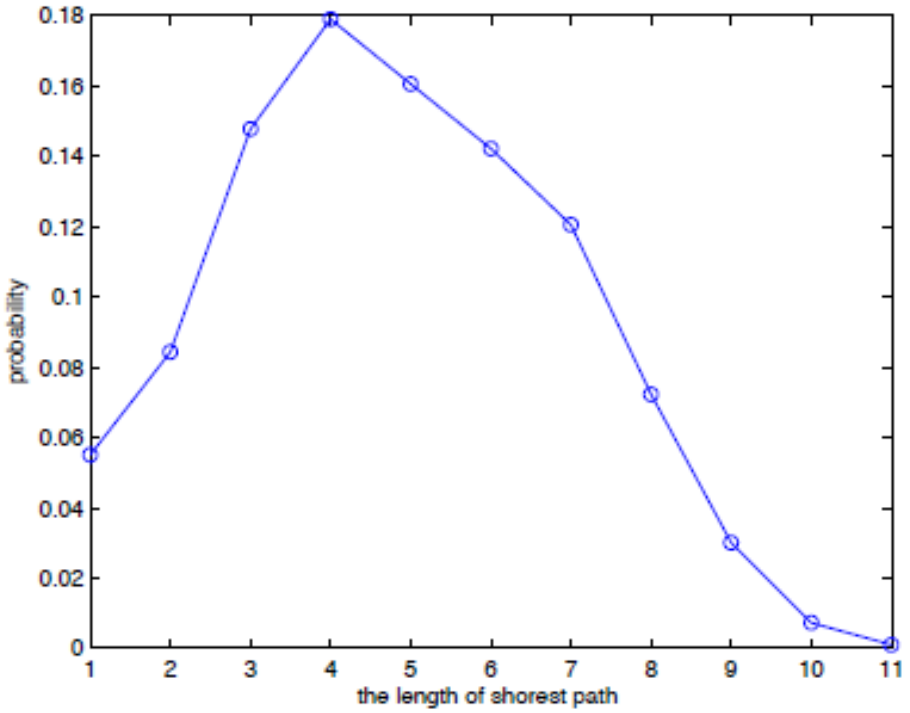
(3) Clustering Coefficients

$$C = \frac{1}{N} \sum_{i} C_i \qquad C_i = \frac{E_i}{k_i(k_i - 1)/2}$$

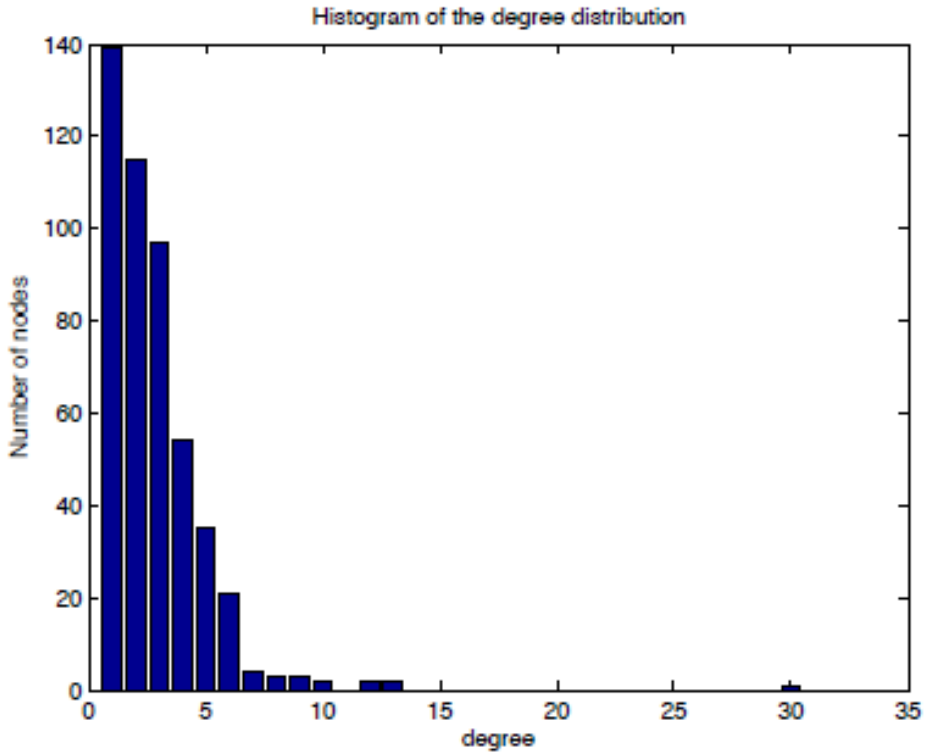
where N is the number of nodes in the network, k_i and E_i are the degree and number of the links among the neighbors of node *i* respectively.

(4) Betweenness

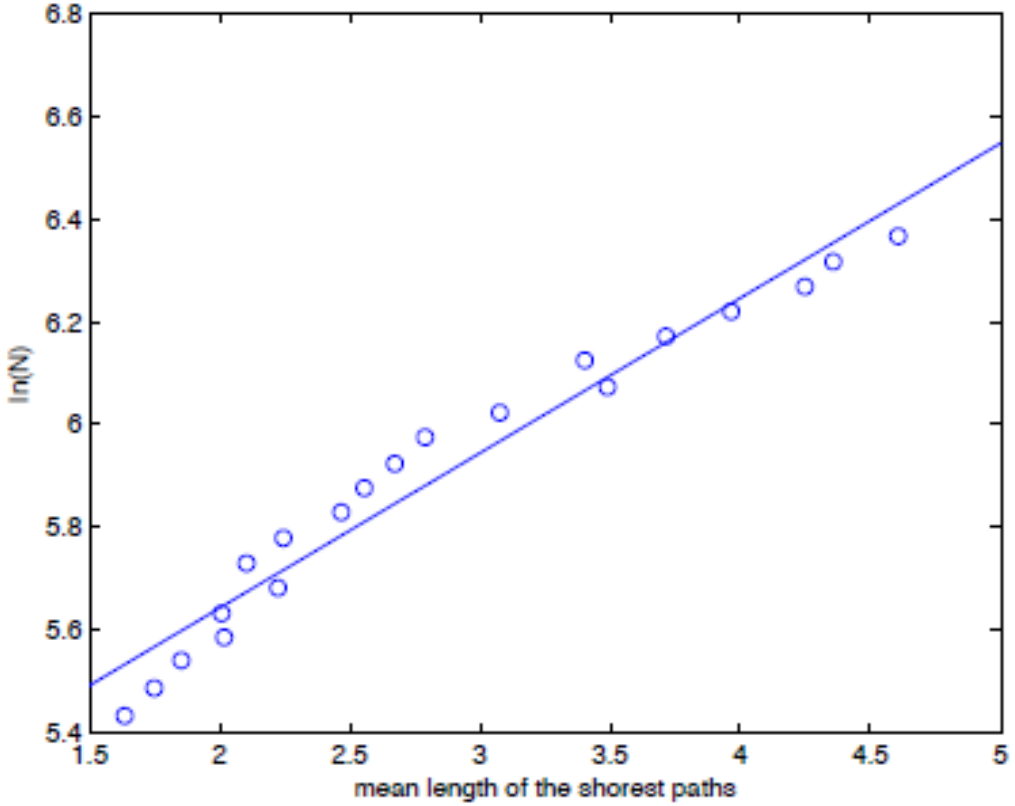
Most companies with high betweenness are located in developed areas.



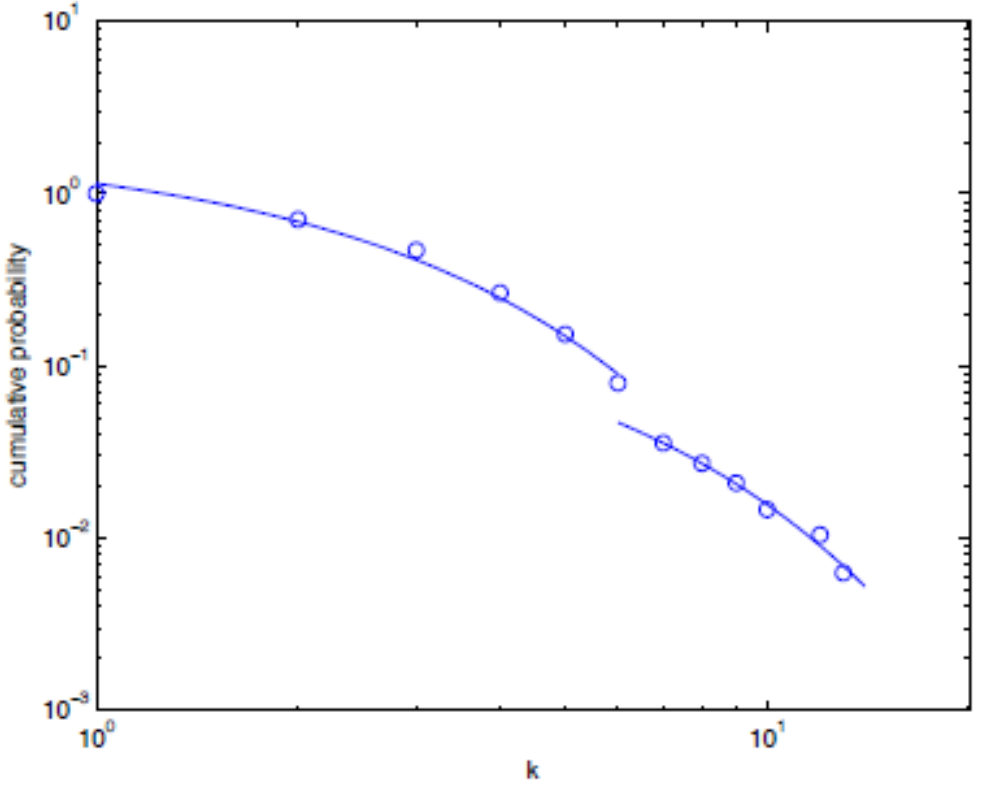
Probability density function of the shortest paths between two nodes of the Chinese VC network



Degree distribution of the Chinese venture capital network



A plot of *ln(N)* vs. $\bar{l}$ for the Chinese VC network. The open circles are the empirical data points and the blue line is its linear fit, $\ln N = 5.04 + 0.3\bar{l}$.



Cumulative degree distribution of the VC network in China. The empirical data gives

$$p = \begin{cases} e^{-0.39k+5.50}, k < 7 \\ e^{-0.27k-1.41}, k \geq 7 \end{cases}$$

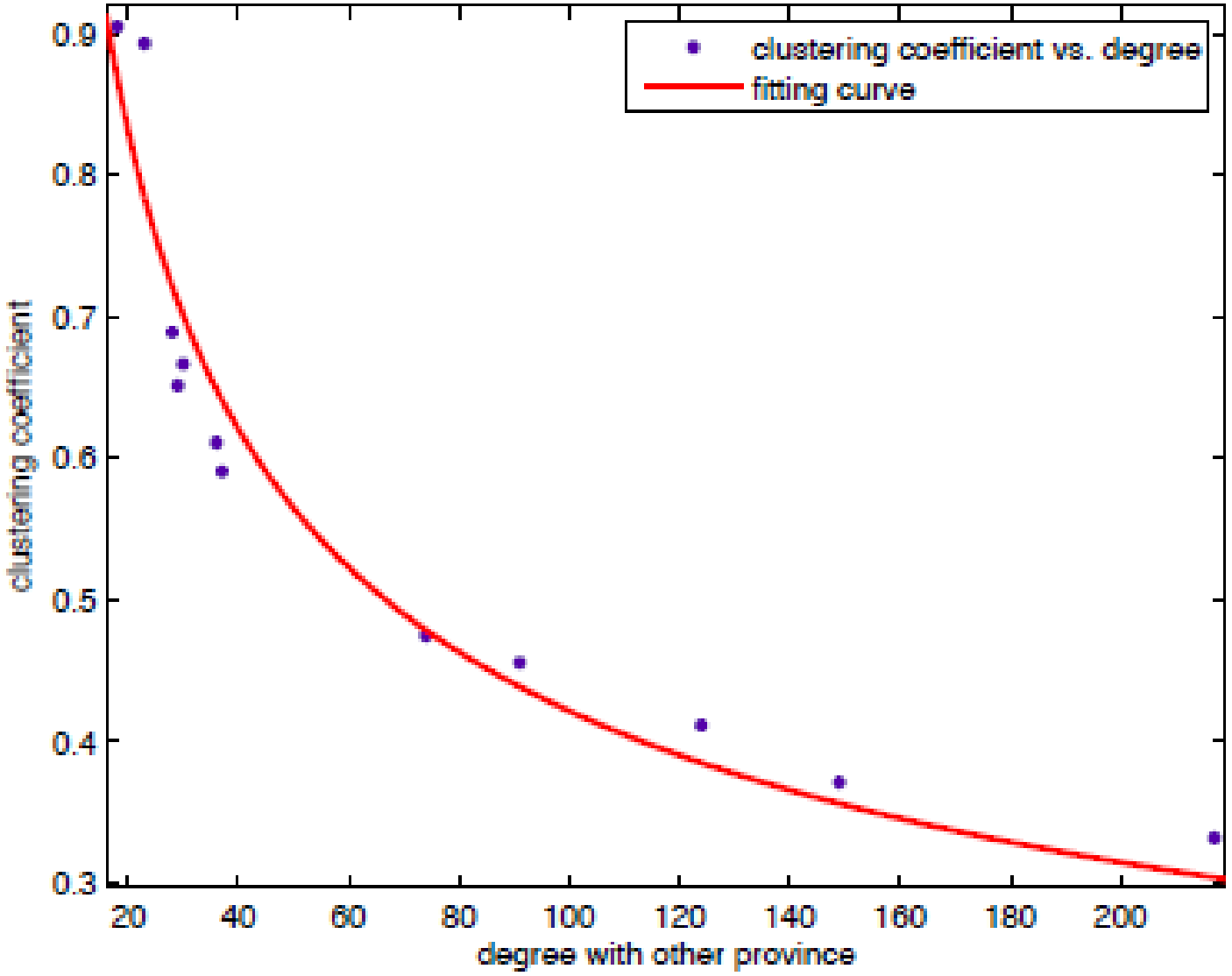
where p is the cumulative probability density function of the degree distribution.

Betweenness and its correlation with node degree

Node degree and district location of 20 venture capital companies with highest betweenness.

Venture capital company	District	Betweenness	Node degree
ShenZhen capital group	Guangdong	4.851	30
IER venture capital Co., Ltd.	Guangdong	1.926	6
ShenZhen OFC investment management Ltd.	Guangdong	1.855	8
ShenZhen CDF-capital company limited	Guangdong	1.532	6
Cowin capital	Guangdong	1.502	13
Guosen Hongsheng	Guangdong	1.497	12
National development creation capital management	Jiangsu	1.198	5
Borong investment	Guangdong	1.076	7
Green investment	Henan	1.012	9
China resources SZITIC trust Co., Ltd.	Guangdong	1.003	13
Fuhai Yintao venture capital	Guangdong	0.855	12
Nanchang venture capital	Jiangxi	0.839	3
Zhongbi fund	Shanghai	0.632	10
Jiangsu high-tech investment group Co., Ltd.	Jiangsu	0.557	8
Guangzhou Haihui growth of venture investment center	Guangdong	0.557	9
Shenzhen Tsinghua leaguer venture capital Co., Ltd.	Guangdong	0.549	5
CSM	Beijing	0.397	7
Shenzhen Hongling venture e-commerce	Guangdong	0.32	6
Hengxiang investment	Guangdong	0.32	10
GTJA innovation investment Co., Ltd.	Beijing	0.32	6

A linear fit gives $y = 0.78x - 1.17 \times 10^{-16}$, where y and x denote betweenness and degree respectively.



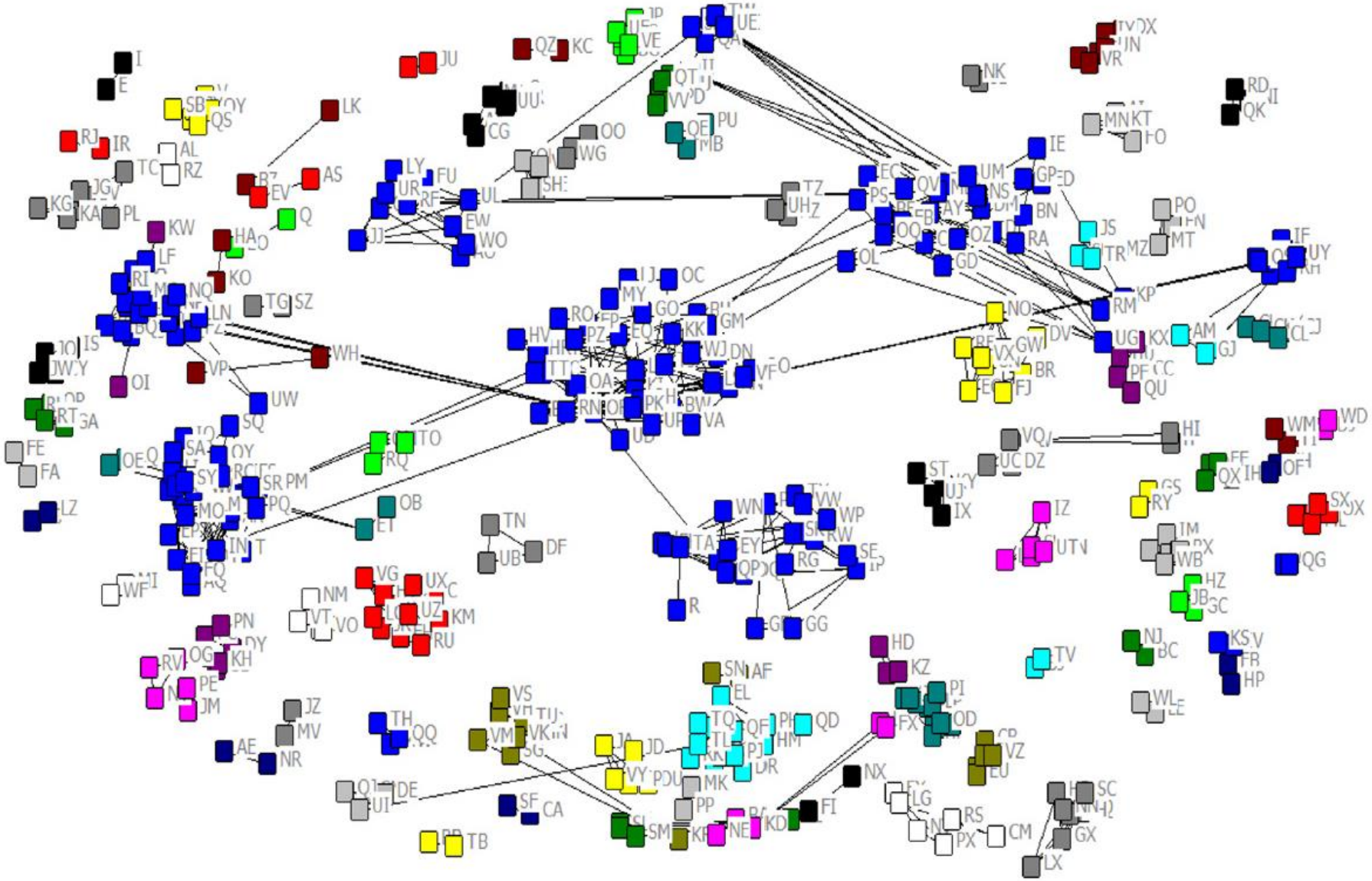
Clustering coefficient of a province/district as a function of its out-degree. One can fit the data points with an approximate power law function $y = 2.94x^{-042}$

Community detection of venture capital networks

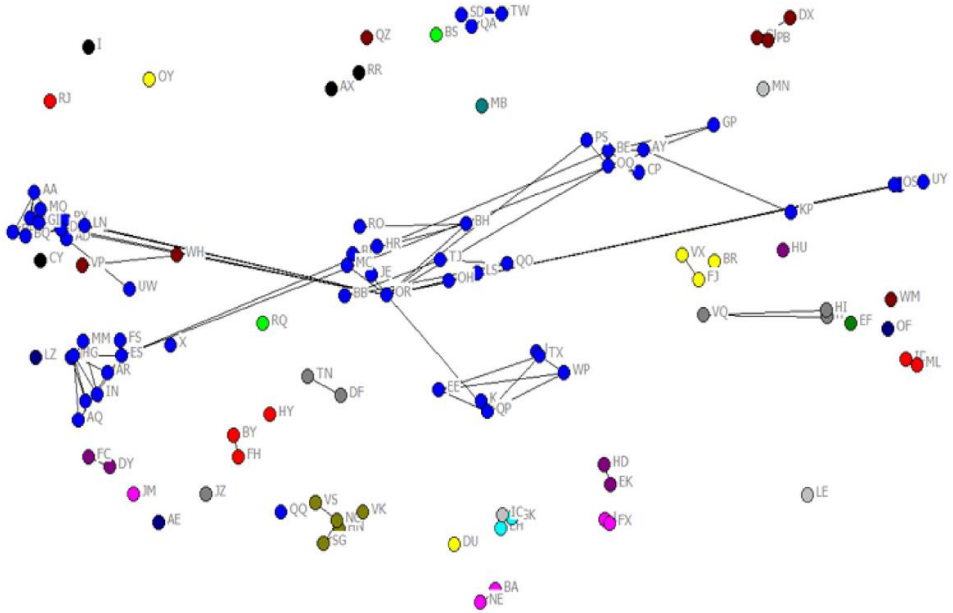
We here adopt the *Girvan–Newman* method in the community detection of the Chinese venture capital network. Recall that the algorithm is generally stated as follows:

1) Compute the betweenness scores for all edges in the network;
2) Remove the edges with the highest betweenness scores;
3) Recalculate the betweenness scores for the network after the removal of edges;
4) Repeat steps 2 and 3 until the whole network is divided into *N* components.

Y.H. Jin et al., *Topological Properties and Community Detection of Venture Capital Network: Evidence from China*, Physica A442(2016)300-311.

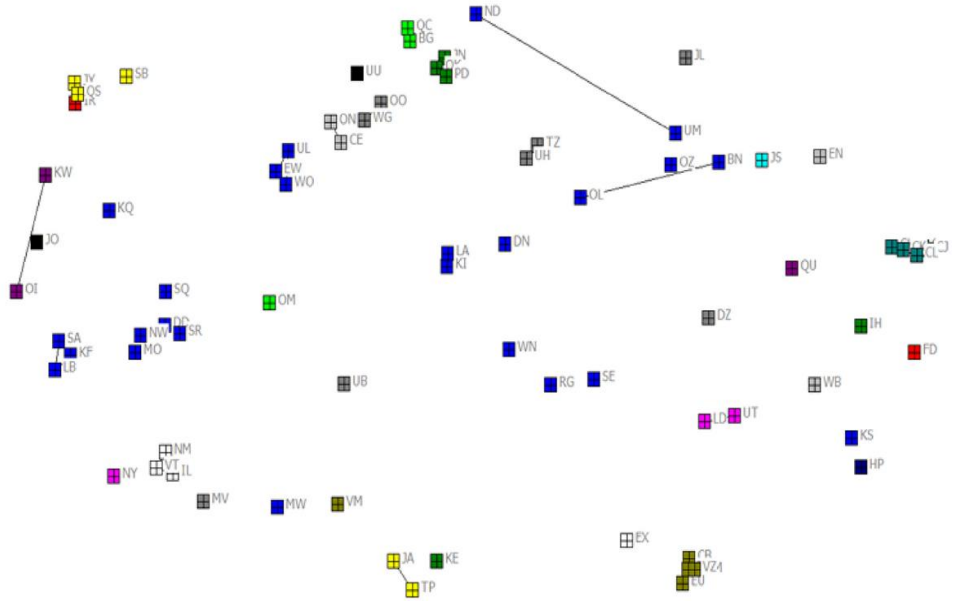


Community structure of the Chinese VC network using Girvan–Newman algorithm. The largest community has 38 nodes, and the smallest ones only comprise two nodes. The number of communities which consist of nodes more than 5 is 22.

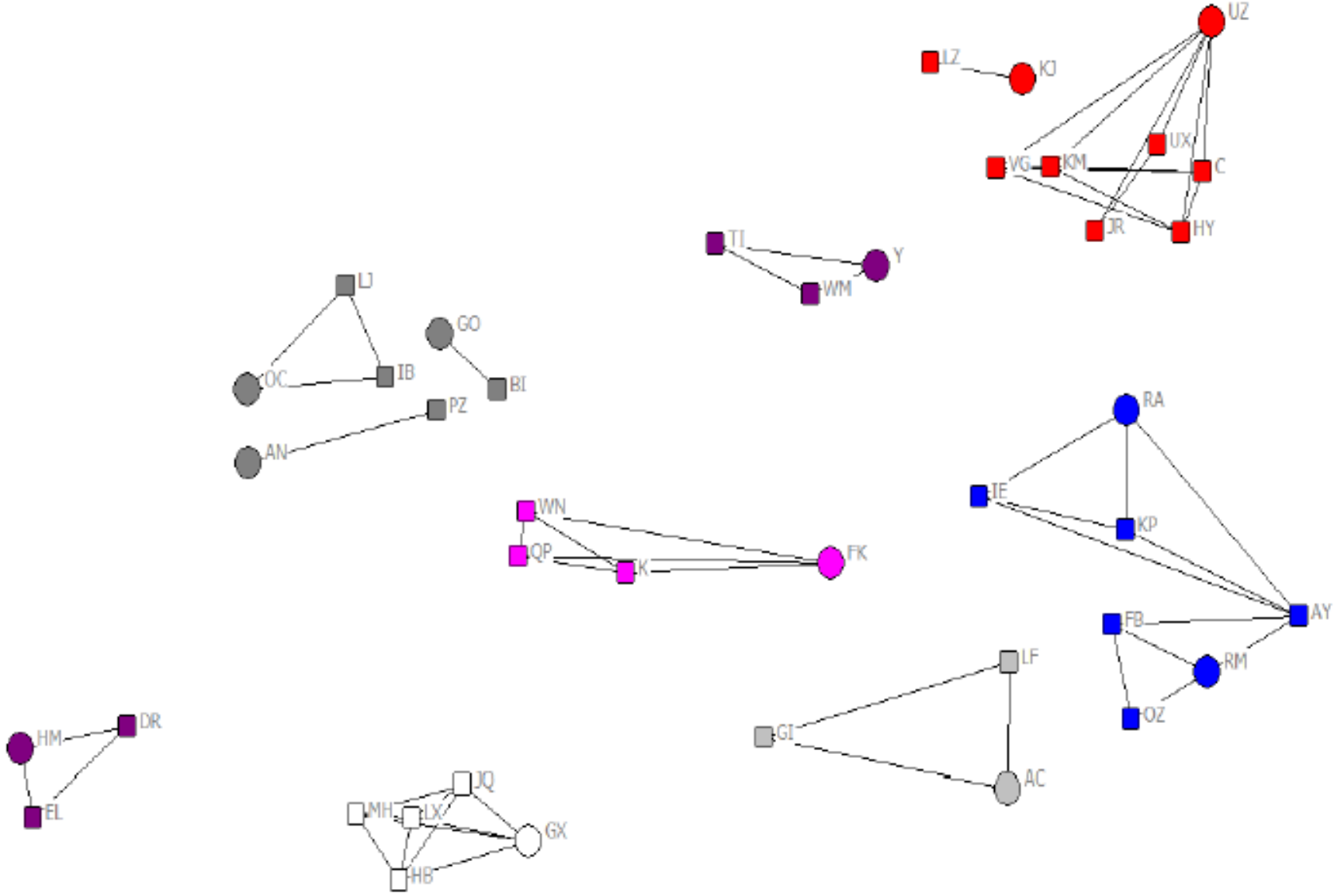


(a)

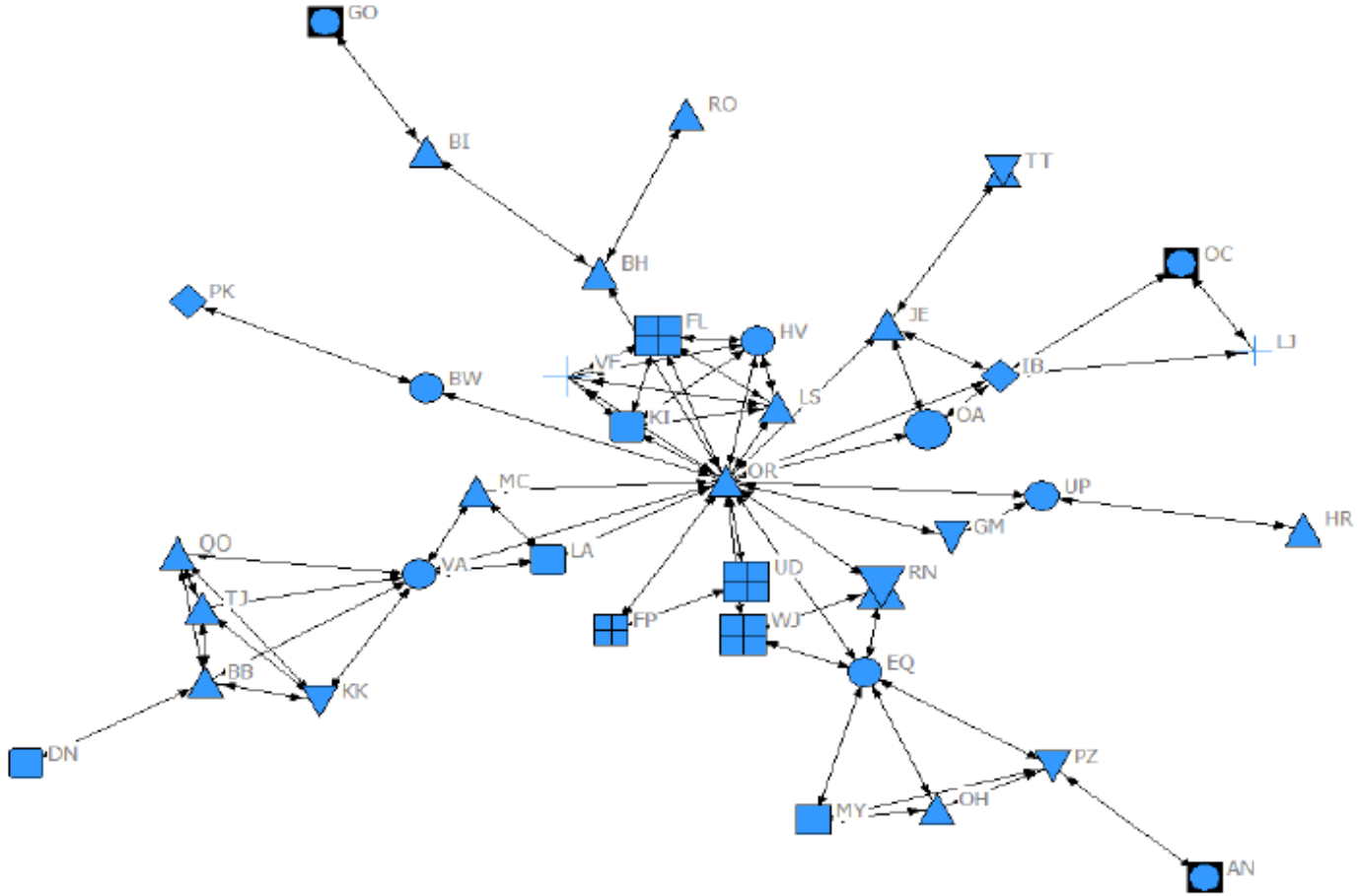
The VC network of
(a) Guangdong and,
(b) Shanghai.



(b)

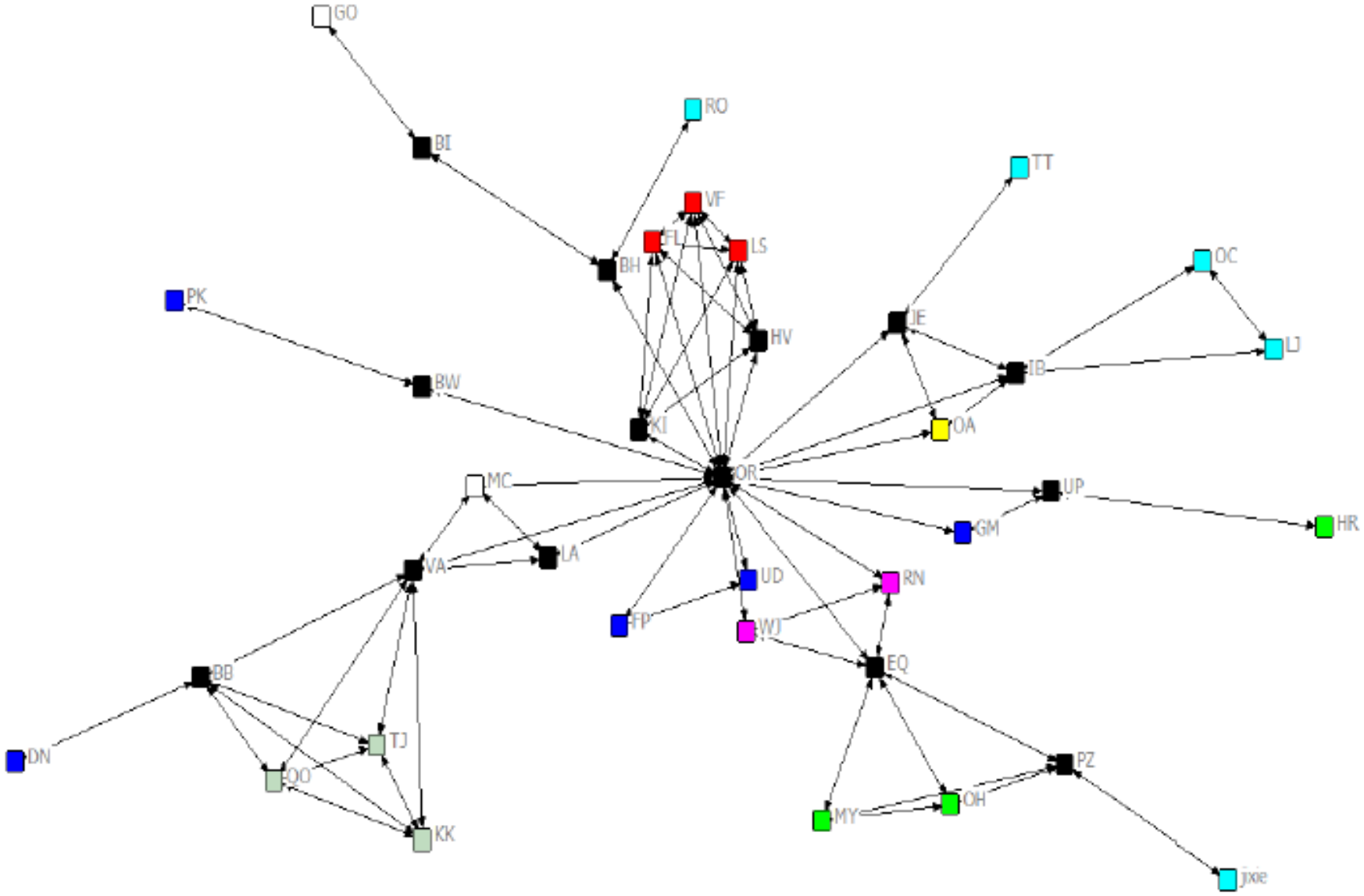


Connection of Hunan's venture capital companies and their ego-networks. Circles represent nodes in Hunan.



District location of VC networks largest community

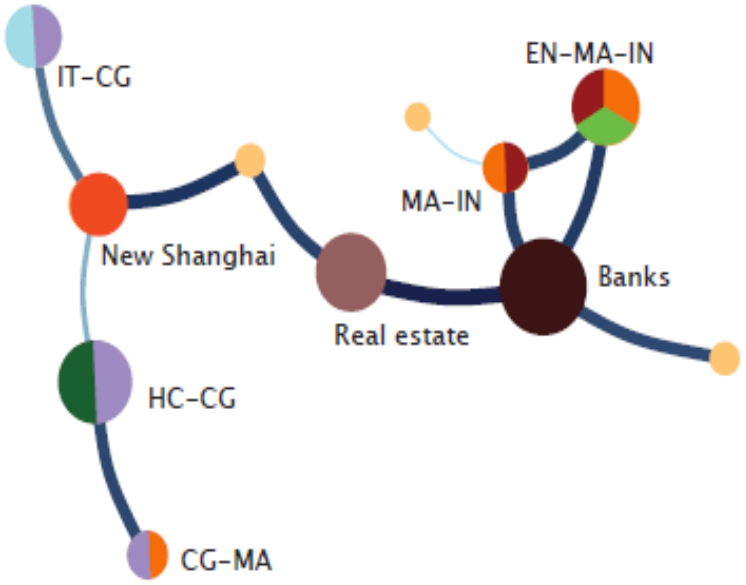
District	Number of nodes	Color	District	Number of nodes	Color
Beijing	5	Circle	Jiangsu	2	Diamond
Fujian	1	Large size circle	Liaoning	1	Plus
Guangdong	12	Uptriangle	Shandong	1	Thing
Hainan	1	Square	Shanghai	3	Rounded square
Henan	1	Box	Sichuan	1	Large size plus
Hubei	3	Downtriangle	Tianjin	1	Large size thing
Hunan	3	Circlebox	Zhejiang	3	Large size box



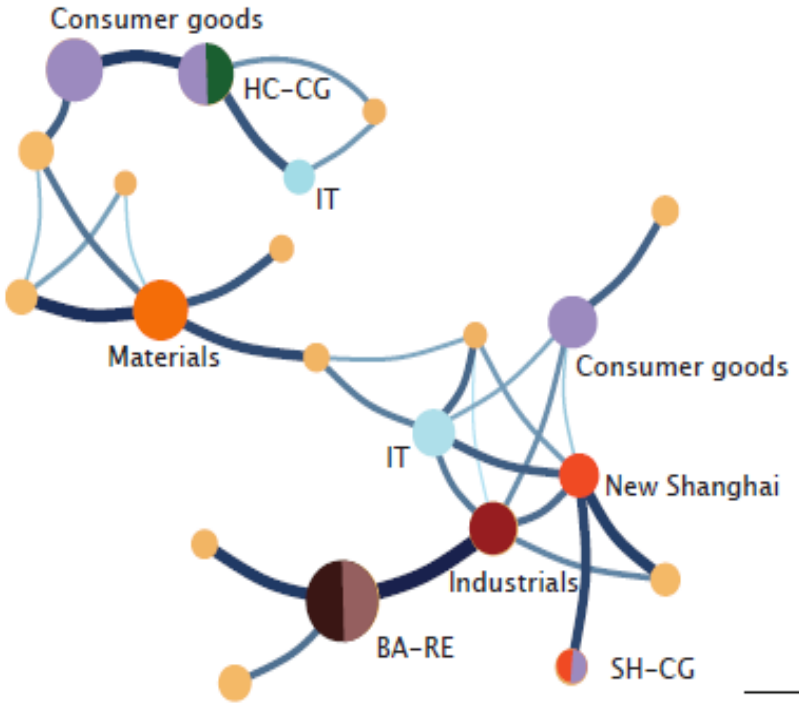
Invested industry distribution of the largest community (Different colors indicate different industries, a node in black indicates that it invests in more than one industry)

Case Study II:
Dynamical Structure of Stock Communities

Dynamic Structure of Stock Communities



(a)



(b)

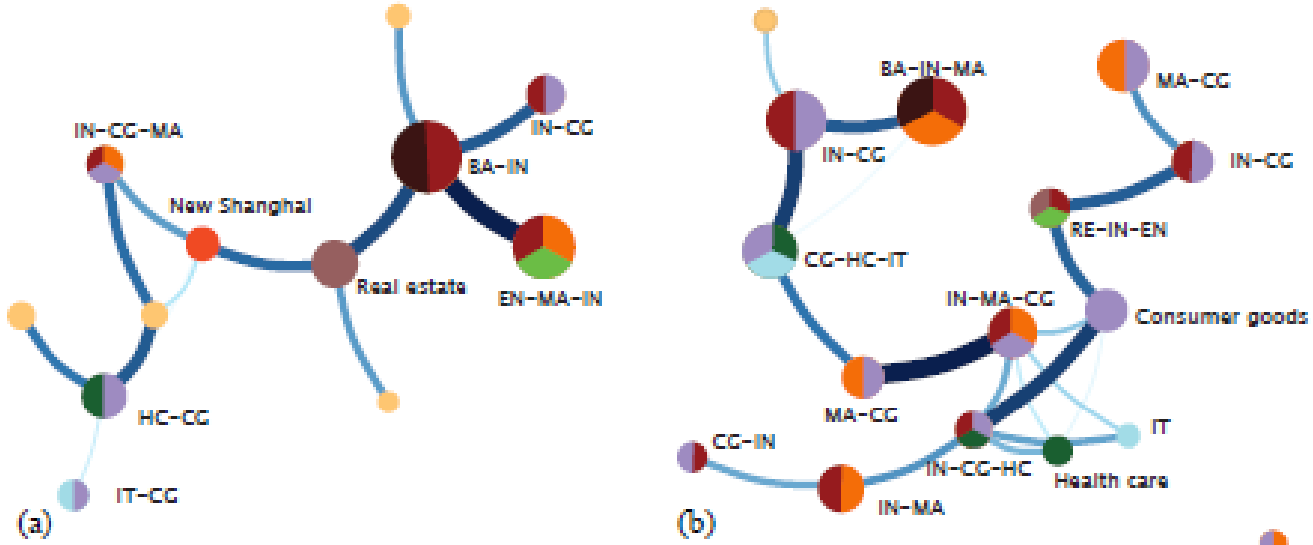
Community structures of the PMFG graphs for (a) returns and (b) turnover rates, using the daily records from October 8, 2007 to March 31, 2015.

"Dynamic structure of of stock communities: A comparative study between stock returns and turnover rates", Li-Ling Su et.al., EPJB 2017.

The industry sectors are categorized based on the Global Industry Classification Standard (GICS). The basic information includes the name and code of the industry sector, and the number of chosen stocks from each sector.

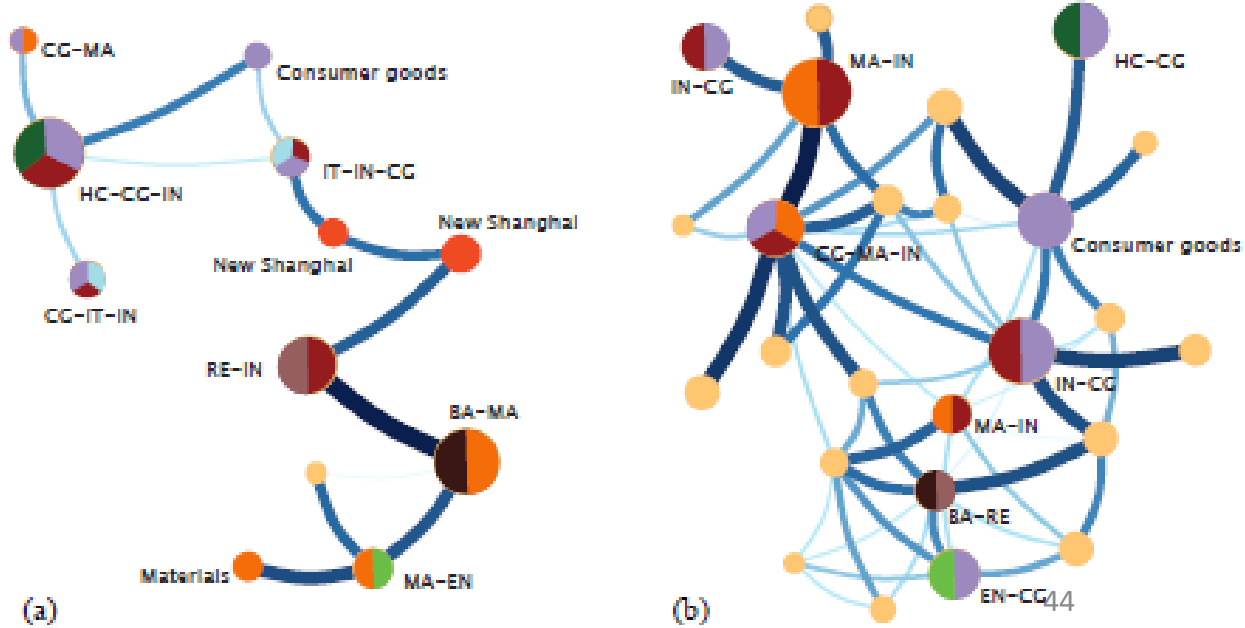
Name	Code	Number of stocks
Energy	EN	11
Materials	MA	60
Industrials	IN	82
Consumer goods	CG	86
Health care	HC	30
Banks	BA	11
Real estate	RE	26
Information technology	IT	27
Utilities	UT	17

Dynamic Structure of Stock Communities

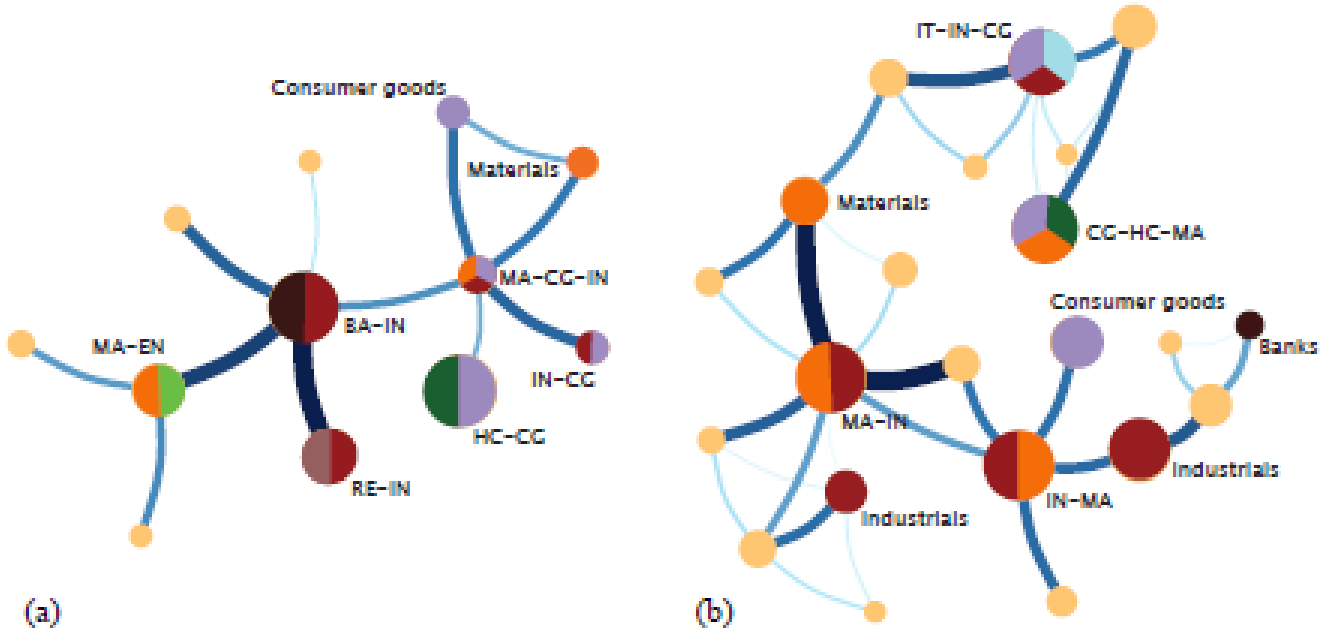


Community structures of the PMFG graphs for (a) returns and (b) turnover rates in the sub-period from October 8, 2007 to March 31, 2009.

Community structures of the PMFG graphs for (a) returns and (b) turnover rates in the sub-period from April 1, 2009 to September 30, 2010.

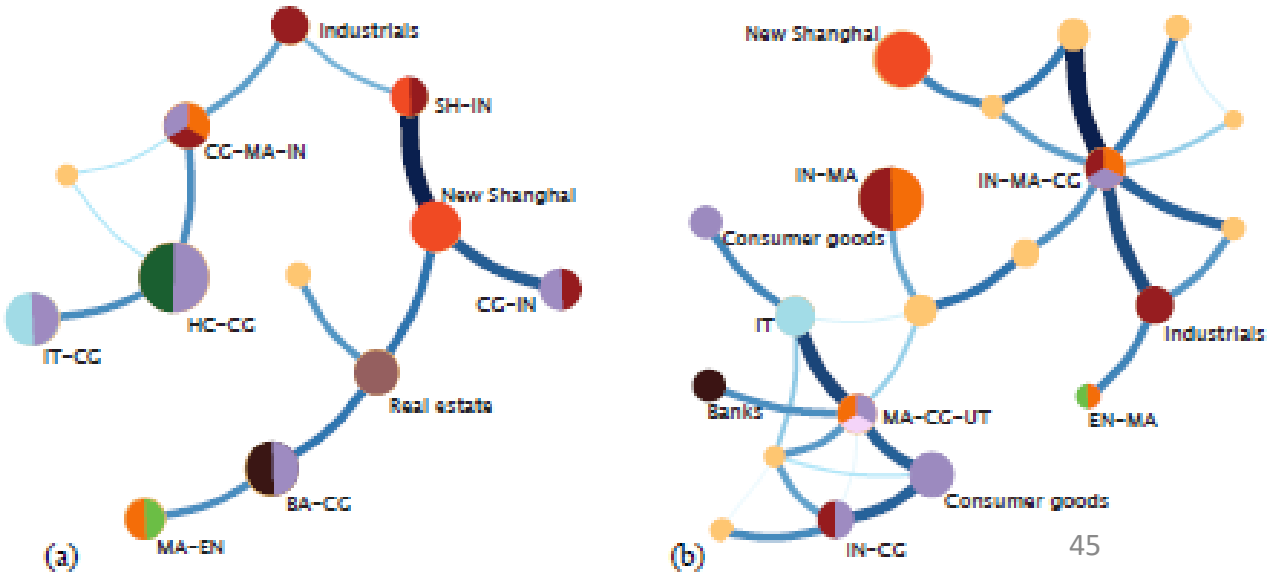


Dynamic Structure of Stock Communities

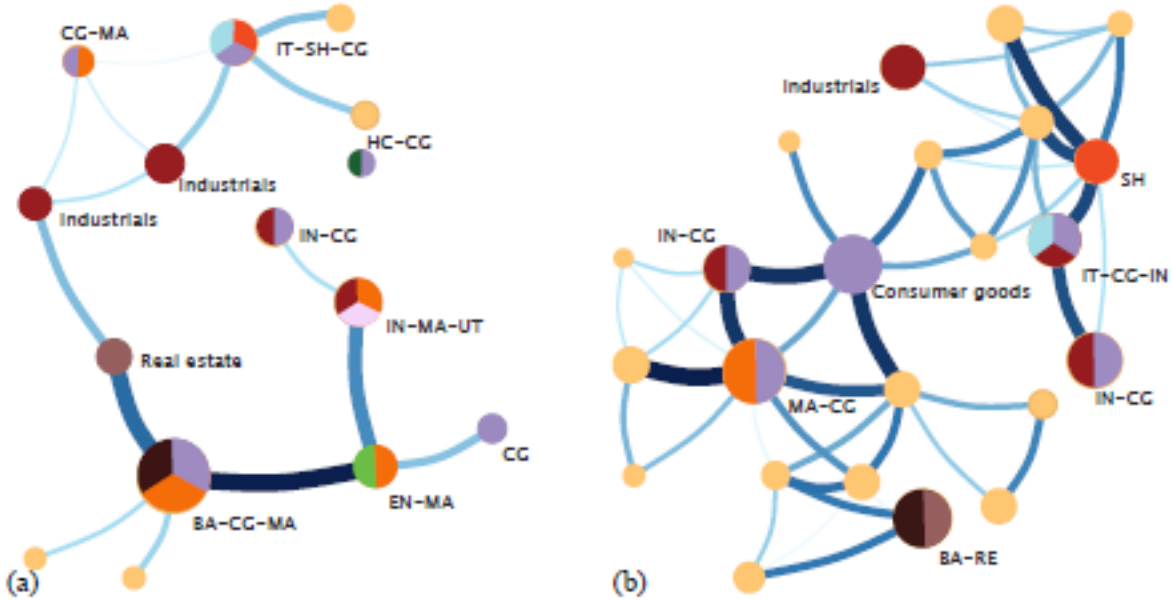


Community structures of the PMFG graphs for (a) returns and (b) turnover rates in the sub-period from April 5, 2012 to September 30, 2013.

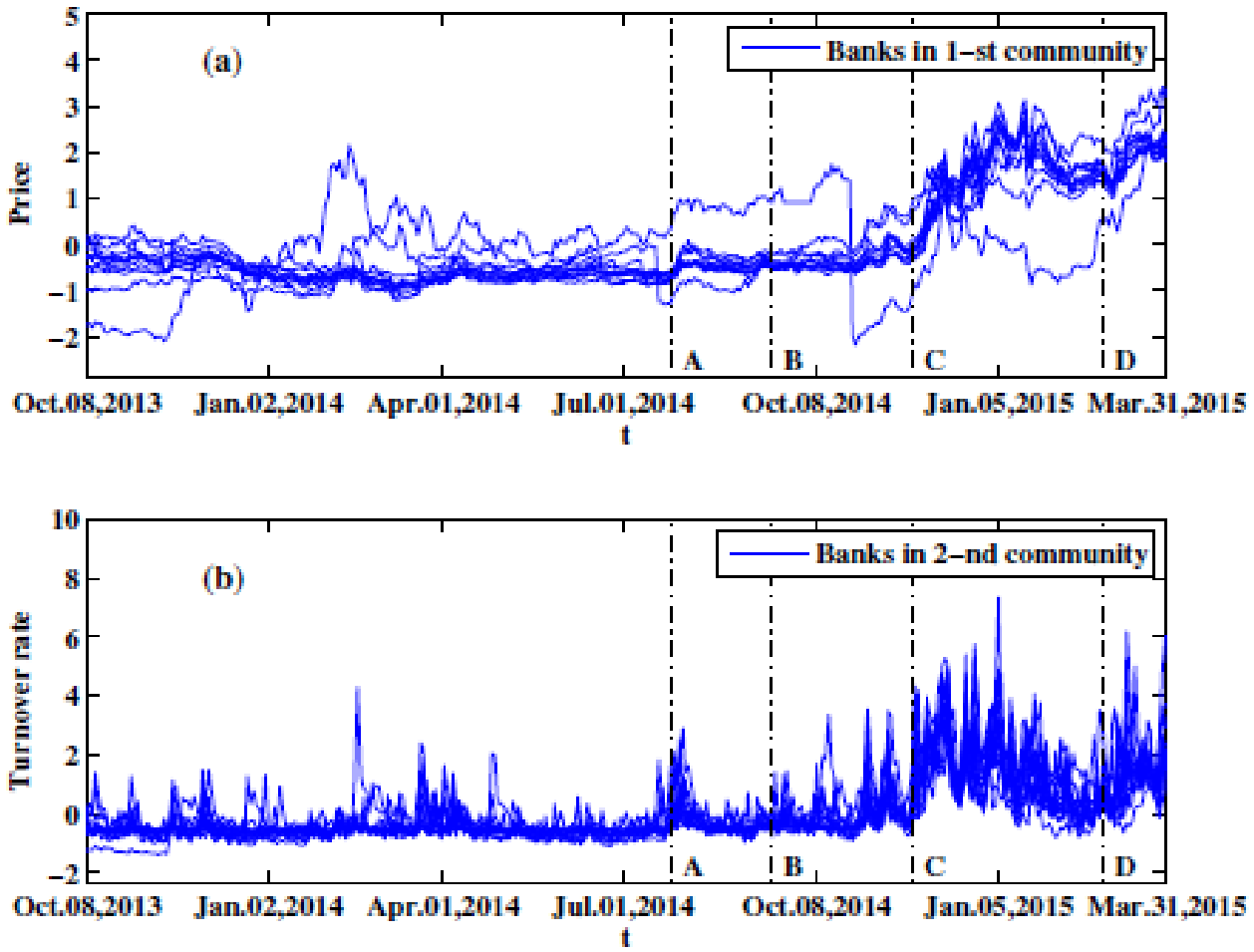
Community structures of the PMFG graphs for (a) returns and (b) turnover rates in the sub-period from October 8, 2010 to March 30, 2012.



Dynamic Structure of Stock Communities



Community structures of the PMFG graphs for (a) returns and (b) turnover rates in the sub-period from October 8, 2013 to March 31, 2015.



(a) Evolutions of prices for stocks from the banks sector in the largest community of the graph for returns. (b) Evolutions of turnover rates for stocks from the banks sector in the 2nd largest community of the graph for turnover rates. All subgraphs are plotted by using data in the sub-period from October 8, 2013 to March 31, 2015.