Web信息处理与应用

实验二：Community Detection

Datasets

数据集	节点	边数	社区数目
College football	115	613	12
Books about US Politics	105	441	3
Com-DBLP	2960	9264	4

Ground truth of the first 2 are given.

Dataset1: College Football

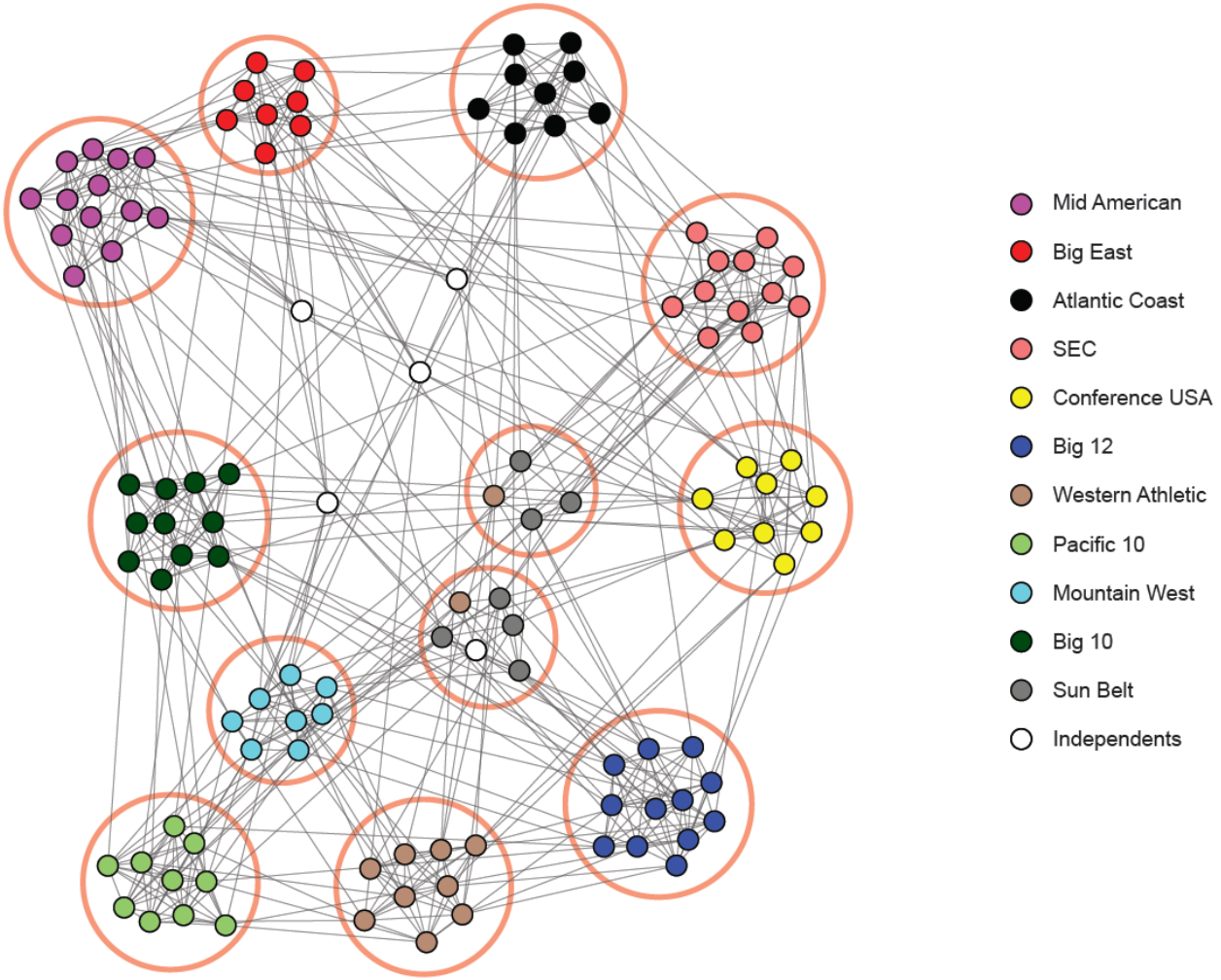
- 2000年秋季美国大学橄榄球比赛收集的数据。节点表示每个橄榄球队，两节点的相连的边表示该赛季对应的两个橄榄球队进行过比赛。数据集一共包含115个橄榄球队，并且分为12个小组进行比赛，这12个小组分别是：

0 = Atlantic Coast	6 = Mid-American
1 = Big East	7 = Mountain West
2 = Big Ten	8 = Pacific Ten
3 = Big Twelve	9 = Southeastern
4 = Conference USA	10 = Sun Belt
5 = Independents	11 = Western Athletic

- 一个赛季内，每个球队平均和组队的约7个不同的队伍进行比赛，同时和组外的约4个不同队伍比赛，组外的比赛队伍的选择并没有统一的规律，可以看成是随机选择。

- 12 communities

Dataset1: College Football



Dataset2: Books about US Politics

- 亚马逊上售关于美国政治类书的一个数据集。节点表示书，边表示有至少一个用户都购买了边上两点对应的两本书。Community Detection的过程可以看成将书细分为“自由派（liberal）”、“中立派（neutral）”和“保守派（conservative）”的过程。

Dataset3: com-DBLP

- The DBLP computer science bibliography provides a comprehensive list of research papers in computer science. We construct a co-authorship network where two authors are connected if they publish at least one paper together. Publication venue, e.g, journal or conference, defines an individual ground-truth community; authors who published to a certain journal or conference form a community.

Dataset4: Facebook-Egonet

- 来自Facebook的一个真实社交网络
- 稀疏的无向无权图
- 4039个节点，88k条边

Algorithm1: Girvan-Newman

```
function clustering = girvannewman(A, k)
```

Repeat until no edges are left:

- Calculate betweenness of edges (*O(mn)*, or $O(n^2)$ on a sparse graph, with breadth-first-search)
- Remove edges with highest betweenness

Algorithm2: Average Link + Jaccard Similarity

```
function clustering=alinkjaccard (A, k);
```

- Jaccard Similarity $Jaccard(\mathbf{v}_i, \mathbf{v}_j) = \frac{|N_i \cap N_j|}{|N_i \cup N_j|}$
- Implement average link agglomerative clustering with Jaccard similarity

Algorithm3: Ratio Cut

`function clustering=rcut(A, k);`

1) Pre-processing
   - Construct a matrix representation of the graph (*D-A*)
2) Decomposition
   - Compute eigenvalues and eigenvectors of the matrix
   - Map each point to a lower-dimensional representation based on smallest *k* eigenvectors (*Vr*: *nxk* matrix)
3) Grouping
   - Assign points to *k* clusters, by running *k*-means on the new representation (`kmeans(Vr, k)`)

Algorithm4: Normalized Cut

`function clustering=ncut(A, k);`

1) Pre-processing
   - Construct a matrix representation of the graph ($D^{-\frac{1}{2}}(D-A)D^{-\frac{1}{2}}$)
2) Decomposition
   - Compute eigenvalues and eigenvectors of the matrix
   - Map each point to a lower-dimensional representation based on smallest *k* eigenvectors (*Vn*: *nxk* matrix)
3) Grouping
   - Assign points to *k* clusters, by running *k*-means on the new representation (`kmeans(Vn, k)`)

Algorithm5: Modularity Maximization

```
function clustering=modularity(A, k);
```

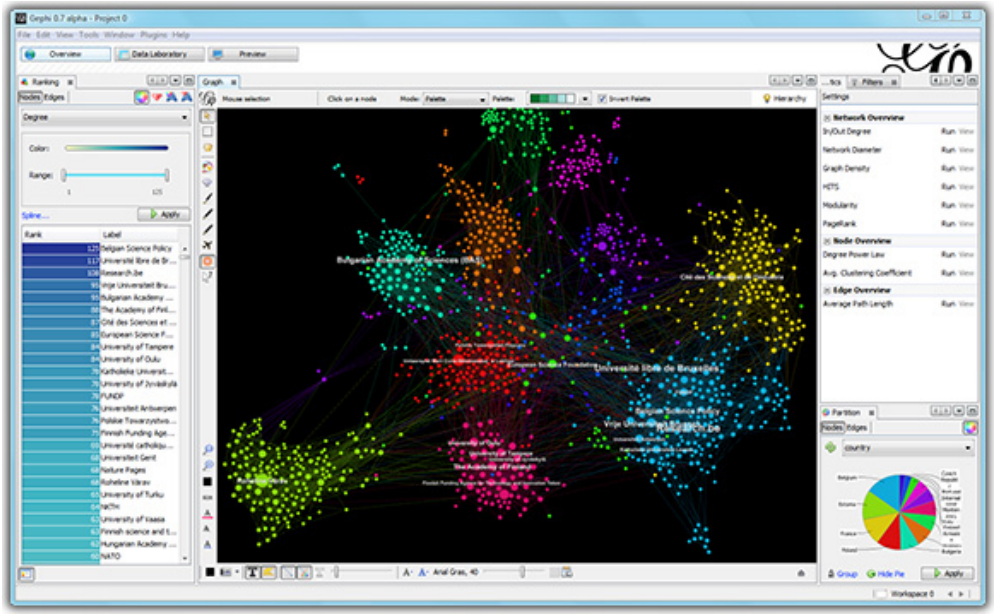
1) Pre-processing
   - Construct a matrix representation of the graph ($B = A - \mathbf{d}\mathbf{d}^{\top}/2m$)

2) Decomposition
   - Compute eigenvalues and eigenvectors of the matrix
   - Map each point to a lower-dimensional representation based on biggest *k* eigenvectors (*Vm*: *nxk* matrix)

3) Grouping
   - Assign points to *k* clusters, by running *k*-means on the new representation (`kmeans(Vm, k)`)

Report

- Discussion on algorithms, analysis of results
- Visualization
  - Gephi



- Advanced discussion (Bonus)

Notes

- Eigen computation

```
>> help eigs
 EIGS  Find a few eigenvalues and eigenvectors of a matrix using ARPACK
    D = EIGS(A) returns a vector of A's 6 largest magnitude eigenvalues.
    A must be square and should be large and sparse.

    [V,D] = EIGS(A) returns a diagonal matrix D of A's 6 largest magnitude
    eigenvalues and a matrix V whose columns are the corresponding
    eigenvectors.
```

```
>> help eig
 EIG    Eigenvalues and eigenvectors.
    E = EIG(X) is a vector containing the eigenvalues of a square
    matrix X.

    [V,D] = EIG(X) produces a diagonal matrix D of eigenvalues and a
    full matrix V whose columns are the corresponding eigenvectors so
    that X*V = V*D.
```

- Sparsity

```
>> help sparse
 SPARSE Create sparse matrix.
    S = SPARSE(X) converts a sparse or full matrix to sparse form by
    squeezing out any zero elements.

    S = SPARSE(i,j,s,m,n,nzmax) uses the rows of [i,j,s] to generate an
    m-by-n sparse matrix with space allocated for nzmax nonzeros.  The
    two integer index vectors, i and j, and the real or complex entries
    vector, s, all have the same length, nnz, which is the number of
    nonzeros in the resulting sparse matrix S .  Any elements of s
    which have duplicate values of i and j are added together.

    There are several simplifications of this six argument call.
```