

858L Homework #2

Due: October 19, before class.

Modularity. Let $G = (V, E)$ be a simple, undirected graph with adjacency matrix A . That is $A_{uv} = 1$ if edge $\{u, v\} \in E$ and 0 otherwise. Let k_u be the degree (number of neighbors) of a vertex u . If we partition the vertices of G into sets M_1, \dots, M_ℓ , the *modularity* of this partition is

$$q(G, M_1, \dots, M_\ell) = \sum_{u \sim v} \left(A_{uv} - \frac{k_u k_v}{2m} \right), \quad (1)$$

where the sum is over pairs of nodes that are in the same module.

Part A. Install GLPK (downloadable from <http://www.gnu.org/software/glpk/>). Write an integer program in the MathProg language that will find a partitioning M_1, \dots, M_ℓ of maximum modularity. Your MathProg program should be in a file called `modularity.mod`. You should assume that the vertices of the graph are labeled $0, \dots, n$.

Your integer program should output a line of the format `# modularity = 0.0`, where the `0.0` is replaced by the maximum modularity of the graph. Following this line, it should output all the pairs of vertices u, v (where $u < v$) that are in the same community. For example, the output could look like:

```
# modularity = 0.44
1 2
1 3
1 4
2 3
2 4
3 4
5 6
5 7
6 7
```

The above is a fake example, just to illustrate the formatting.

Part B. Write a Python program that will read a graph G (in the same format as Homework #1) and write a MathProg data file that can be input to the integer program of Part A. Your python program should be in a file called `modularity.py`.

In other words, the user should be able to find the maximum modularity partition of a graph by issuing the following commands:

```
python modularity.py graphfile.edg output.data
glpsol -m modularity.mod -d output.data -y result.out
```

Multiway Cut Model:

```
param N integer > 0; # number of nodes
param A integer > 0; # number of annotations

set ZEROONE := {0, 1};
set V := 1 .. N; # the set of vertices
set E within V cross V; # the edges, set by the data
set Annot := 1 .. A; # set of annotations
set FIXED within V cross Annot cross ZEROONE; # triples of fixed variables

var Xn{u in V, a in Annot} binary;
var Xe{(u,v) in E, a in Annot} binary;

maximize monochrom: sum{(u, v) in E, a in Annot} Xe[u,v,a];

subject to upper1 {(u,v) in E, a in Annot}: Xe[u,v,a] <= Xn[u,a];
subject to upper2 {(u,v) in E, a in Annot}: Xe[u,v,a] <= Xn[v,a];

subject to fixed {(u,a,x) in FIXED}: Xn[u,a] = x;
```

Multiway Cut Example Data:

```
param N := 5;
param A := 3;
set E := (1,2) (2,3) (1,3) (1,4) (3,5) (2,5) (4,5) (3,4) ;
set FIXED := (1,1,1) (5,2,1) (4,3,1) ;
```