

Graph Algorithms

Lecture 11

October 18, 2005

Graphs

- $G=(V,E)$
 - V = vertices, $|V|=n$
 - E = edges
 - $\{u,v\}$ – undirected graph
 - (u,v) – directed graph
- Features:
 - Degree of a node
 - Cycles
 - Reachability
 - Connectedness
 - Any vertex is reachable from any other vertex



Graph representation

- Incidence list:
 - For each vertex v , we have a list $Adj(v)$ of vertices u such that (u,v) is an edge

| | |
|---|-----|
| 1 | 5,2 |
| 2 | 1,3 |
| 3 | 2,4 |
| 4 | 3,5 |
| 5 | 4,1 |

- Adjacency matrix:
 - $n \times n$ binary matrix A
 - $A[u,v]=1$ if and only if (i,j) is an edge
 - Denote $A[u,v]$ by a_{uv}

| | | | | |
|---|---|---|---|---|
| | 1 | | | 1 |
| 1 | | 1 | | |
| | 1 | | 1 | |
| | | 1 | | 1 |
| 1 | | | 1 | |

Graph problems

- Connected components (undirected graphs)
- Cycle detection
- Graph clustering

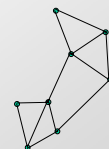
Connected components (CC)

- Assume adjacency list representation
- We use
 - An array $Mark[u]$, initialized to 0
 - An array $CC[u]$
- How do we find all nodes that belong to the connected component on node 1 ?
 - Run $Search(1,1)$, where
 - Search(u,v):**
 - If $Mark[u]=1$ then Exit
 - $Mark[u]=1, CC[u]=v$
 - For u' in $Adj(u)$ do $Search(u',v)$
- How do we search for all connected components ?
 - For $v=1$ to n do $Search(v,v)$

Graph clustering

- Partition V into clusters $C_1 \dots C_k$ so that edges “tend” not to cross cluster boundaries
- For simplicity, focus on $k=2$
That is, partition V into C and $V-C$
- Want to minimize
 - The sum of the edges in
 $Cut(C) = \{ (u,v) : u \in C, v \notin C \}$ ← Minimum Cut
 - The ratio

$$\frac{|Cut(C)|}{|C|}$$
 ← Sparsest Cut
 subject to $|C| \leq |V|/2$
 - ...or subject to $|C|=|V|/2$ ← Bisection



Formulation of bisection

- Want to find C , $|C|=|V-C|$, which minimizes $\text{Cut}(C)$
- Let A be the adjacency matrix of G
- We represent C using indicator variables $c(v)$, which for any vertex v :
 - $c(v) = 1$ if and only if $v \in C$
 - $c(v) = -1$ if and only if $v \notin C$

- Goal: minimize

$$\sum_{u,v} (1 - c(v)c(u)) a_{uv}$$

subject to $\sum_u c(u) = 0$, $c(u) \in \{-1, 1\}$

- Alternatively, maximize

$$\sum_{u,v} c(v)c(u) a_{uv}$$

subject to $\sum_u c(u) = 0$, $c(u) \in \{-1, 1\}$

Relaxation of bisection

- We relax constraints on $c(u)$
 - Instead of $\{-1, 1\}$, they can be arbitrary real numbers
 - Need to add a constraint on their magnitude, e.g., $\sum_u c(u)^2 = n$
- Altogether, we get
 - Maximize $\sum_{u,v} c(v)c(u) a_{uv}$
 - Subject to:
 - $\sum_u c(u) = 0$
 - $\sum_u c(u)^2 = n$
- In the matrix notation:
 - Maximum $c^T A c$
 - Subject to:
 - $c^T 1 = 0$
 - $\|c\|^2 = n$
- Can be solved by computing eigenvalues of A
 - In particular, if all nodes have the same degree, then c is the 2nd eigenvector of A
 - In Matlab:
 - $[V,D] = \text{eig}(A)$
 - c is the 2nd last column of V

Example

- A graph with 50 nodes, split into C and $V-C$
 - Probability of any two nodes having an edge is:
 - 1/3 if nodes are in the same cluster
 - 1/11 if nodes are in different clusters
 - Compute $c(u)$ for all u
 - Order nodes clockwise in the increasing order of values $c(u)$
- Images removed due to copyright restrictions.

Rounding

- We get real vector c
- Need to convert to $\{-1, 1\}$
- Method:
 - Order nodes by $c(u)$
 - First $n/2$ nodes get -1
 - Last $n/2$ nodes get $+1$

