Graphs: Quick reference

vishvAs vAsuki

May 1, 2012

Contents

Part I

Notation

Part II

Themes

Discover better, more realistic graph generators. Discover more realistic algorithms for various tasks like finding shortest path.

1 Characterization of research effort

There is both experimental and theoretical work.

Part III

Graphs and their basic properties

2 Graph G=(V,E)

E annotated with weights. If G unweighted, all weights are 1.

2.1 Vertex properties

Degree of v: $deg(i) = \sum_j e_{i,j}$. Neighbors of v: $\Gamma(v)$.
Measure size of $A\,\{V\}$: $|A|$ or $vol(A) = \sum_{i \in A} deg(i)$.

2.2 Associated objects

2.2.1 Special vertex sets

Node cover of G=(V,E): subset of V which touches all e in E.
Proper vertex coloring; chromatic number.
Independent set of verteces: don't share edge: a clique in $\bar{G}$.

2.2.2 Walks

A sequence of edges (e_i) such that e_i shares an end-point with e_{i+1}.
A walk may have a cycle. If it does not have a cycle, you have a path.
Also see random walks.
Alternating walks are defined and studied in combinatorial optimization problems over graphs; in these sequences even and odd edges are colored differently.

2.2.3 Cut

k-way cut: $cut((V_i))$ is a partitioning of V into k parts. Cutsets is a set of edges which, when removed from G, divides V into k partitions.

2.2.3.1 Weight of a cut

Weight of a cut is the sum of weights of edges in the cutset.
Minimum 2-way cut is a cut with the minimum weight. This is useful in partitioning nodes in a graph.

2.2.4 Subgraphs

A general subgraph: G'=(V', E'). Subgraph induced by $A\{V\}$. Connected components of a graph. Clique: a complete subgraph.

2.2.4.1 Spanning trees

Spannning tree. MST spanning a certain node set N: Aka Steiner tree. MST spanning with support over node set N: Aka Group Steiner tree.

2.3 Subtypes

Multigraphs: multiple edges allowed.

2.3.1 Tree

G sans cycle. Forest F: set of trees.

2.3.2 Biconnected graph

2 paths between any node pair.

2.3.3 Based on degree

d-regular G: $\forall v : |N(v)| = d$. Complete graph K_n.

2.3.4 Perfect graph G

Chromatic number = size of the largest clique in G.
A graph is perfect iff its complement is perfect.

2.3.4.1 Bipartite graph

E = cutset. Complete bipartite graph: All v in A has edge to all u in B. Complete bipartite graph $K_{i,j}$. Can have A vs B adj matrix M. Thence get usual adj matrix for G: $\left[\begin{smallmatrix} 0 & M \\ M^T & 0 \end{smallmatrix}\right]$.

2.3.4.2 Chordal graph G

Aka triangulated graph. Every cycle in G has a chord. G has a junction tree iff it is chordal.

2.3.5 Planar graphs

(Kuratowski) G planar iff K_5 and $K_{3,3}$ are in G.

2.4 Directred graph, networks

Digraph or directed graph. Networks: digraph with edge weights.

2.4.1 Directed acyclic graphs (DAG)

Very useful in designing algorithms: can do recursion easily.

2.4.1.1 Topological numbering t of nodes

Number nodes so that if there is a path from u to v, then $p(u) < p(v)$. Cyclic graphs don't have a topological numbering.

2.4.2 Singly connected directed graph

Useful in probabilistic graphical models.
A tree underlies the graph: only 1 undirected path between any node pair.

2.5 Generalizations

2.5.1 Hypergraphs

Edges connect k-sets of verices, not just pairs.

2.6 Properties

2.6.1 Hop plot

For h, let g(h) be number of node pairs with path $\leq h$. Hop plot plots this.

2.6.2 Diameter

Diameter of G. q effective diameter: q fraction of $V \times V$ have path length $\leq d$.

2.6.3 Single source (s) shortest paths

Consider a weighted graph with weight $e(a, b)$ between two nodes. Suppose that you want to find the shortest path to every vertex $v \in V$ from s.
One can use a bottom-up programming approach. (Dijkstra) Start with current node $c = s$. Initially set $d(s) = 0$ and, $\forall v \neq s : d(v) = \infty$.
Define $f(c)$: For every $v \in \Gamma(c)$, do the update: $d_{t+1}(v) := \min(d_t(c) + e(c, v), d_t(v))$. Also, simultaneously update the 'backpointer' to point to c (representing the optimal subsolution) if necessary.
Do $f(c) \forall v \neq s$.
Time: $O(n^2)$ in case of a complete graph.

2.7 Associated matreces

Edge incidence matrix J ($m \times n$); for weighted G: edge $e_{i,j}$ terminii (i, j) are marked $\pm\sqrt{e_{i,j}}$.
Vertex incidence/ adjacency matrix W: presence of edge $e_{i,j}$ indicated by weight at $W_{i,j}, W_{j,i}$.
Connectivity matrix A^∞.
Degree matrix: D = diag(deg(i)).

2.7.1 Graph Laplacian of no-self-loop undirected graph

L = (D-W). $L = JJ^T$. As L is $|V| * |V|$, it is an operator on the functions with domain V.

2.7.1.1 +ve semidefiniteness

L is symmetric, +ve semidefinite: $x^T Lx = x^T JJ^T x$. So $\lambda \geq 0$.

2.7.1.2 Smoothness of vectors from quadratic form

$x^T Lx = x^T Dx - x^T Wx = \sum_{i,j} W_{i,j}(x_i^2 - x_j x_i) = 2^{-1} \sum_{i,j} W_{i,j}(x_i - x_j)^2 = \sum_{e_{i,j} \in E} e_{i,j}(x_i - x_j)^2$. This is a measure of the degree of oscillations/ smoothness among x, where edges occur.

2.7.1.3 Eigenvectors

$L1 = 01, \therefore \lambda_1(L) = 0$; so L singular.
If G has c connected components: $\lambda_1 = ..\lambda_c = 0$: construct ev with 1 in the appropriate spots!

2.7.1.4 Smoothness of ev

Take ev x. Then $x^T Lx = \sum_{e_{i,j} \in E} e_{i,j}(x_i - x_j)^2$ measures smoothness of x where edges occur in the graph. But, ew are stationary points of $R(x) = x^T Lx/(x^T x)$. So, ev corresponding to lower ew tend to be smoother.

Smoothening functions Consider the subspace spanned by the p bottom (smooth) ev. Any function on V, ie a $|V|$-dim vector can be projected on to this subspace in order to smoothen it according to the graph structure. Labelling of nodes is an example of such a function.

Applications This property is useful when using ev x for classification of nodes - one doesn't want neighboring nodes to have disparate values in x. This is useful in both spectral clustering and label propogation in semisupervised learning.

2.7.2 Normalized graph Laplacian

Take $N = I - D^{-1/2}WD^{-1/2}$: this is the normalized version, $D^{-1/2}LD^{-1/2}$ of L, using the normalized adjacency matrix $D^{-1/2}WD^{-1/2}$.
$N \succeq 0$ as $L \succeq 0$, ie $x^T Dx - x^T Wx \geq 0 \forall x$: taking $D^{1/2}x = y$, see that $\forall y : y^T Ny \geq 0$.

2.7.2.1 Normalized adjacency matrix has norm 1.

As $y^T y - y^T D^{-1/2}WD^{-1/2}y$
≥ 0, see that $1 \geq \left\|D^{-1/2}WD^{-1/2}\right\|_2$; Also, using $y = D^{1/2}1$, get
$\left\|D^{-1/2}WD^{-1/2}\right\|_2 = 1$.
Another way to see this: $D^{-1/2}WD^{-1/2}$ is obtained by a similarity transformation to $D^{-1}W$, which has ew in the range [-1, 1] due to Gerschgorin thm, and which has $\sigma_m ax = |\lambda_m ax| = 1$ using the ev 1.

2.7.2.2 Quadratic form: Normalized smoothness measure

$y^T Ny = y^T D^{-1/2}LD^{-1/2}y =$
$\sum_{(i,j)\in E} W_{ij}(\frac{y_i}{\sqrt{d_{ii}}} - \frac{y_j}{\sqrt{d_{jj}}})^2$: from the form $x^T Lx$ being a smoothness measure.

Punishes deviation between $\{y_i\}$ corresponding to edges emanating from high degree vertices less.

3 Metrics and similarity measures on V

Similarity measures between u and v are inverses of distance metrics.

3.1 Based on paths and random walks

- (shortest path).

3.1.1 Katz measure

$sim(u,v) = \sum_{l=1}^{\infty} \beta^l |paths_{u,v}(l)|$. Matrix of scores, $K = \sum_{i=1}^{\infty} \beta^i A^i$.

3.1.1.1 The damping parameter

$K = (I - \beta A)^{-1} - I$ if the sum converges. Use $A = U\Lambda U^*$: EW decomposition, with λ_i ordered in descending order. $\sum_{i=1}^{\infty} \beta^i A^i = U(\sum_{i=1}^{\infty} \beta^i \Lambda^i)U^*$ does not converge $\forall \beta \geq 0$ and multiplication by ∞ is not well defined. Condition for convergence: $\beta\lambda_1 < 1$.
But, for $\beta > 1$ the intuition of weighting longer paths less does not hold.

3.1.2 Variants of Katz

Similarly can use $e^{-\beta A}$.
Truncated Katz usually used: $\sum_{l=1}^{k} \beta^l A^l$: $O(ln^2 nz(A))$ op instead of $O(n^3)$ inverse finding.

3.1.3 Hitting time

$-h_{u,v}$; normed by stationary distribution: $-h_{u,v}\pi_v$: to take care of skewing of hitting time due to large π_v. - Commute time: $-h_{u,v} - h_{v,u}$; stationary distr normed: $-h_{u,v}\pi_v - h_{v,u}\pi_u$.

3.1.4 Rooted PageRank

Random walk can get lost in parts of graph away from u and v; so do random resets and return to u with probability a at each step.

3.1.5 SimRank

A recursive definition: 2 nodes are similar to the extant that they are joined to similar nodes. $sim(x,x) = 1; sim(x,y) = \frac{\sum_{a\in\Gamma(x)}\sum_{b\in\Gamma(y)} sim(a,b)}{|\Gamma(x)||\Gamma(y)|}$.

3.2 Common neighbor based

Common neighbors: $|\Gamma(u) \cap \Gamma(v)|$: same as taking inner product of rows in adj matrix M corresponding to u and v. (Jaccard): $\frac{\Gamma(u) \cap \Gamma(v)}{\Gamma(u) \cup \Gamma(v)}$: pick a feature at random, see probability that it is a feature of both u and v. (Adamic/ Adar): $\sum_{z \in \Gamma(u) \cap \Gamma(v)} \frac{1}{\log |\Gamma(z)|}$: greater wt to rare features present in both.

4 Random graphs: $G_{n,p}, G_{n,|E|}$

4.1 G(n,p)

A static model: every edge is present or absent independent of other edges. p controls edge density. As $n \to \infty diam(G) \to 2$. Size of the largest cluster is $O(\log n)$.

5 Expanders

A sparse graph with high connectivity properties. Connectivity quantified as edge expansion or vertex expansion. Let $E'(S)$: edges with exactly one end point in S.

5.1 Edge expansion of G

Aka isoparametric number.
$h(G) = \min_{1 \leq |S| \leq n/2} \frac{|E'(S)|}{|S|}$.

5.2 α vertex expansion of G

$g_\alpha(G) = \min_{1 \leq |S| \leq n\alpha} \frac{|\Gamma(S)|}{|S|}$.

5.3 Other Examples

Most graphs are expanders. K_n has good expansion properties, but it is not sparse.
[**Incomplete**]

6 Clique tree of G

Look at maximal cliques, pick a set of such cliques so that every edge is in some clique, make corresponding clique-nodes, make a tree by adding paths between cliques passing through shared-vertex-set nodes.
Eg: 1-2-3 has clique tree (1,2)-2-(2,3).
G can have multiple clique tree.

6.1 Junction tree property

Take any path in the clique tree between cliques C_1 and C_2; then $C_1 \cap C_2$ appears in every shared-vertex-set node inbetween them.
If a clique tree has it, it is a junction tree. These are useful in inference using graphical models: see probabilistic modelling ref.
Only chordal graphs have a junction tree; but can always add edges to convert non chordal graphs to chordal graphs.

6.1.1 Finding a junction tree

Take all maximal cliques (this is easy for chordal graphs), connect them with edges with certain weights, find maximal weight spanning tree. [**Incomplete**]

6.1.2 Tree width of G

Take t = The size of the maximal clique in the junction tree of G with the smallest maximumal clique size. The tree width = t-1.

7 Relationship with Electric network analysis

Kirchoff current (y) and voltage (x) laws, Ohm's law: edge current sources f: $Ax = b, A^T y = f, y = CAx$. Elimination in A: rows independent if not loop; Formation of spanning tree; so r=n-1. So, left nullspace basis = loops = m-n+1. Implies **Euler Formula**: n - m + loops = 1.

8 Random walks on G

8.1 Markov process with state graph G

Take a Markov chain with probability matrix $P_{u,v} = \frac{1}{deg(u)}$. $\pi_v = \frac{d(v)}{2|E|}$. [**Check**]

8.2 Hitting and cover times

Hitting time $h_{u,v}$: expected time to get to v from u. Commute time: $C_{u,v} = h_{u,v} + h_{v,u}$. $h_{u,v} < 2|E|$ [**Find proof**].
Cover time C(G) = $\max_v$ E[time to hit all nodes starting from v]. $C(G) < 4|V||E|$ [**Find proof**].

8.3 Connection with resistive electric networks

Let every $e \in E$ have resistance 1, thence get n/w N(G). $R_{u,v}$: effective resistance between u, v. $C_{u,v} = 2mR_{u,v}$ [**Find proof**].

Part IV

Determining graph properties

9 Node search

9.1 Searching for a goal node

Problem; initial state; successor function; goal test; path cost; step cost.
The search tree vs state graph; its branching factor. NP hardness. Optimality vs completeness of algorithms. Eg: Route finding problem with a map; TSP.

9.2 Uninformed Search problem

Breath first search; exponential memory demands. When pathcost varies: Uniform cost search.
Depth first search: linear space complexity. Iterative deepening DFS: use diameter of state graph. When pathcost varies: Iterative lengthening search.
Bidirectional search.
Avoid repeated states: open and closed lists of states.

9.3 Informed search

At node x, distance to goal node: d(x). Heuristic functions h(x). Admissible heuristics: $h(x) \leq d(x)$; never overestimate. Consistent heuristics and the triangle inequality. Greedy best-first search. A*.

9.4 Local search/ optimization

Eg: Protein structure prediction. Hill climbing. Statespace landscape: Local minima and shoulders. Random restarts. Simulated annealing. Local beam search. Genetic algorithms.

9.5 Offline vs online search

Eg: Search on internet without a prior index; protein design. Learning in online search (exploration); updating the search tree and heuristic values of nodes.

10 Network flow

Take network G=(V,E) with source s, sink t;
$(u, v) \in E$ has capacity c(u,v). Flow $f : V \times V \to R$ with capacity con-

straints, flow conservation $\sum_u f(u,v) = 0$, skew symmetry: f(u, v) = -f(v, u). Residual capacity of an edge: $c_f(u,v)$. Thence, can get residual network G_f. Augmenting path $s, v_1, v_2..t$: $\forall i, c_f(v_i, v_{i+1}) > 0$.

10.1 Max flow problem

Start with 0 flow. Max flow exists iff $\nexists$ augmenting path. Check for augmented flow; Keep increasing or decreasing flow by small fractions.

10.2 Min cut problem

Reduce to max flow problem.

11 Tree search

11.1 Find minimum spanning tree (MST)

(Kruskal) Start with forest F = V; pick $e \in G$ with least weight; if e connects 2 trees in F, add it to F: else discard e; repeat.
(Prim) Grow a tree T starting from a vertex: at each step, add edge with least wt which brings in new vertex to T.

12 As data structures

See algorithms and data structures survey.

Part V

Evolution of real world graphs

13 Networks in the Real world: Properties

See [1].

13.1 Examples

May be directed or undirected. Citation graph can be considered directed. Coauthorship graphs. Query graphs. Online social network websites

13.1.1 Affiliation networks

A bipartite graph G'=(A,C,E) of actors A, communities C and edges E. Can implicitly define social network amongst A: by folding G'/ forming edges between actors depending on shared memberships in $c \in C$.

13.2 Static patterns

13.2.1 Small diameter

Small world phenomenon in social networks: remarkably short paths connect very unrelated people. Eg: A certain developmental psychologist has Erdos number 3. 6 degrees of separation idea.
Two randomly selected web pages are connected is around .35.
So, effective diameter of the graph is small.

13.2.2 In and out Degree distribution

Heavy tailed distributions. Folks look at shape of the graph, and observe a distribution pattern.
May follow power law distribution over a large degree-range; observed in online social networks, citation graphs. So, number of nodes with degree d $\propto d^{-\beta}$. β may or mayn't vary over time.
Else DGX distribution.
Else log normal distribution.
In-degree distributuion is more skewed, usually.

13.2.3 Scree plot

Plot of ew vs rank using log vs log scale. Approximately obeys power law: You see a straight line.

13.2.4 Clustering coefficient

In social networks, most new edges form to close triangles. Let degree(v)=d. Clustering coefficient $C_d(v) = \frac{\Delta(v)}{\Gamma(v)(\Gamma(v)-1)/2}$, where $\Delta(v) = \triangle$ centered at v: the fraction of potential $\triangle$'s centered at v which actually exist.
$C_d = avg_{v:deg(v)=d}(C_d(v))$. Gets smaller with increasing degree. Follows power law distribution: $C_d \propto d^{-1}$.
Global clustering coefficient $C = avg_v(C_d(v))$. It is a measure of transitivity in the network. C in real networks is significantly higher than for random networks, conditioned on same degree distribution.
Indicates some degree of hierarchial and community structure. The low-degree nodes belong to very dense sub-graphs and those sub-graphs are connected to each other through hubs.

13.2.5 Community structure

13.2.5.1 2 meanings

Communities amongst V observed in real world G, due to homophily. In the internet, many dense bipartitite subgraphs observed (Border).
Communities in the social networking websites often don't correspond to dense subgraphs of the social network.

13.2.5.2 Hierarchical structure

Found in social network, internet: a scale-free self similar structure observed.

13.2.5.3 Fractal or onion structure

Similar structure observed as you keep zooming in on a part of G.

13.2.5.4 Communities in the social networking websites

Group size: Power law distribution. Number of affiliations of nodes of a certain degree: exponential distribution. [3] Group membership: a high fraction of singletons: especially in small groups, decreases with group size. With groups size, highest degree of member nodes increases.

13.2.6 Core-periphery structure

found in internet autonomous systems.

13.2.7 Network motifs

Basic building blocks of complex networks. Frequency of occurance compared to random graphs, function then hypothesized. Graph isomporphism checks feasible only for motifs of size ≤ 5.
Eg: gene regulatory networks.

13.2.8 Microscopic properties

Most (70 to 90%) edges are bidirectional [1]..

13.3 Other properties

Eigenvector components distribution skewed.
Resilience to random node attacks, but suceptibility to high degree hub node attacks.
Stress.

13.4 Temporal evolution patterns

13.4.1 Densification

Polynomial densification: $|E(t)| \propto |V(t)^{\alpha}|$ with $\alpha > 1$. So, avg degree increases with time.

13.4.1.1 Relationship with power law degree distribution

$\beta \in (1,2) \implies \alpha = 2/\beta$.
$\beta > 2, \alpha \implies \beta(n) = \frac{4n^{\alpha-1}-1}{2n^{\alpha-1}-1}$ changes in a certain way over time.
Both observed in actual graphs [1].

13.4.2 Shrinking diameter

Diameter shrinks over time, asymptotically levels off.
Densification alone doesn't not result in this (experiment on $G_{n,p}$). Densification with particularly evolving degree sequence result in this. Nothing special about edge attachments: make random graphs with same degree sequences. [1]

13.4.3 Node arrival function

N: time $\to \{0,1\}$. High variation among networks: exponential to sublinear.

13.4.4 Edge initiation process

13.4.4.1 Node lifetime L

After a certain time, node does not form new edges. Spike at L=0, as many people join the network and never visit the networking website again. Best modelled by $Pr(L = l) = ke^{-lk}$: exponential distribution.

13.4.4.2 Age

Edge initiation rate seems to stay constant accross ages.
Graph shows spike at age=0, as many people join the network and never visit the networking website again.

13.4.4.3 Time gap G between edges

G follows power law with exponential cutoff. $p(x) \propto x^{-\alpha}e^{-x\beta}$. α stays constant with time, but β grows linearly.

13.4.5 Edge (u, v) destination selection process

Most (30 to 60%) new edges close triangles: local. Pr(v is k hops away) $\propto e^{-ck}$; Pr(certain v: k hops away) is doubly inverse exponential as number of verteces k hops away itself grows exponentially. [1]

For the creation of the first edge: $Pr(degree(v) = d) \propto d^t$, where t = 1 or 0.9 etc..

14 Generative models for real world graphs

14.1 Applications

Finding patterns and abnormalities in real world networks.
Making either fine or coarse grained hypotheses of graph formation process.

14.1.1 Making Extrapolations

What will G look like 10 months later?
So, algorithmic benefits of the model important: How easy is it to calculate paths? Eg: A may be queries, C may be retrieved documents, maybe we want to find the closest commercial/ advertisement-carrying queries, so want to find shortest path to commercial query in the network of queries.

14.1.2 Graph sampling

Deduce information about the graph by sampling a small number of nodes: if the temporal properties of the statistics which result from this sampling.

14.2 Evaluation of models

Does the model produced have desired global static and temporal properties?
Are the fine grained node arrival and edge formation processes modelled as a time, or do nodes arrive, form edges and stop?
How likely is the generation of a certain real-world network, or of a network with similar properties by the random processes in the model? Is the edge-formation process likely under it?
How useful is it? Is graph sampling easy to determine its properties?
Is it easy to extrapolate its properties analytically? Can you calculate the likelihood of a model given a real world graph G? If so, ye can pick the max likelihood model and extrapolate what G looks like n years later.

14.3 Achieving various properties

14.3.1 Strategies

Look at models which achieve certain properties, even if they don't satisfy other properties; combine their best flavor to produce a good model.
Or observe closely microscopic nw evolution properties: edge initiation, edge destination selection, lifetime etc.., mimic them, see if it produces desired global properties.

14.3.2 Power law

Power laws often appear in combination with self-similar fractal structures.
Naive power law generator: edges come in, pick m from power law, randomly create m edges.

14.3.2.1 Hypotheses

Power law for in-degree is achieved by some flavor of 'preferential attachment' method.
Heavy tails emerge if nodes try to optimize connectivity under resource constraints. So may be humans are engineering things, like the internet, language.
Eg: Take a language with n words, cost of word with length j is log j; this leads to power law distribution on word length.
Criticism: But then, monkey typing a keyboard with n letters and a spacebar achieves power-law distribution on word-length.

14.3.2.2 Other factors

A model based on exponential node lifetime distribution and power law with exp cutoff distr for time gap between edges yields power law out-degree distr.

14.3.3 Acheiving good community structure

Copying model acheives good community structure.

14.3.4 Edge destination selection

Most edges (u, w) close triangles. Triangle closing: many choices for w, what yields best improvement over random choice of w among 2-hop neighbors: choose $v = \Gamma(u)$ by process A; then choose $w = \Gamma(v)$ by process B. Biggest improvement yielded when processes A and B are uniform random choice: bias towards w with more 2-hop paths leading to them.

14.4 Models which don't yield heavy tailed distributions

14.4.1 Random graph generators

(Erdos, Renyi) G(n,p) model. G(n, 0.5) is the uniform distribution over all graphs with $|V| = n$. G(n,N) model. Exhibits phase transition phenomenon. [**Incomplete**]
Don't produce heavy tailed distributions.

14.4.2 Stochastic Adjascency matrix

Take adj matrix, replace 1 and 0 with probabilities p and q. If p=q, get Erdos, Renyi graph.

14.5 Preferential attachment generators

New node arrives; draws number of edges m from a distribution; probability of connecting with u $\propto degree(u)$.
Yields power law in-degree distribution with degree 3, low diameters. 'Rich get richer'.

14.6 Edge Copying model

Node v arrives, picks no of edges m, either picks node v and does this m times: a] connects to random $u \in \Gamma(v)$ or b] with prob β connects to random node. Generates power law distribution with exponent $(1-\beta)^{-1}$.
Leads to good community structure.

14.7 Random walk model

New node comes in, does random walk, for each visited node, create edge with some probability. Generates power law distribution.

14.8 Community guided attachment

Represent recursive structure of communities with a tree T of height H. Take perfectly balanced T; leaves will be the nodes; $Pr((u,v) \in E) \propto c^{-\delta(u,v)}$, where δ is tree distance.
Leads to densification: $E(t) \propto N(t)^a$, as expected. If c near 1, lack community structure; if near 2, get good community struct.

14.8.1 Dynamic community guided attachment

Start with one node. Keep adding levels to a b-ary balanced tree by adding nodes $\{u\}$, out-edges from each u to v with $Pr((u,v) \in E) \propto c^{-\delta(u,v)}$.
Leads to densification: $E(t) \propto N(t)^a$; also to power law distribution for in-degrees.

14.8.2 Defects

Diameter grows slowly. No heavy tailed out degree distr.

14.9 Small world model

Start with lattice; for each edge with prob p, move edge to random node. Low p values tend to yield good community structure.
(Kleinberg) Yields small diameter, easily discoverable paths if long range neighbors chosen in a certain way.

14.10 Forest fire model

Node v arrives, chooses preexisting ambassador node u, forms link to u, picks random x and y, selects x out-links and y in-links from u to $\{w_i\}$, forms out-links to $\{w_i\}$, does the same from each $t \in \{w_i\}$ recursively: the fire spreads.

14.10.1 Good properties

Heavy tailed in-degree: due to 'rich get richer' flavor: highly connected vertices easily reached, irrespective of initial ambassador node.
Community structure, due to 'copying' flavor.
Heavy tailed out-degree: good chance to form many links.
Densification power law: due to 'community guided attachment' flavor: lot of links formed near the community.
Shrinking diameter also observed.
[**OP**]: : Does it have good clustering coefficient distribution?

14.11 Modelling Microscopic network evolution

Parameters: Node arrival fn N(), α, β.
At each time step: check for Node arrival according to N(); every new node samples lifetime and creates first edge according to preferential attachment; nodes $\{u\}$ which are awake create an edge (e, w) by triangle closing by picking random neighbor v and then a random $w \in \Gamma(v)$; they then sample sleep time from distr: $Pr(G = g) = g^{-\alpha}e^{-g\beta}$; repeat.
When a real graph is evolved with this model, it achieves power law degree distribution, expected clustering coefficient vs degree graph, distribution of shortest paths. Densification, shrinking diameter observed by other authors.

14.12 By evolution of affiliation network G' = (A,C,E')

14.12.1 Preferential attachment + edge copying

(Lattanzi, Sivakumar).
Affiliation nw evolution: Pick $d', d'' \in N$; start with affiliation nw G'_0 where each $a \in A$ has min degree d', every $c \in C$ has min degree d". With probability p, evolve A: add an actor a, find a preferentially chosen prototype $a' \in A$, randomly copy d' of its edges. Similarly, with prob (1-p) evolve C.
At each time step, can get social nw G=(A,E) among A by creating an edge (a,b) when they share a community; upon addition of an actor a, can also add edges to s preferentially chosen actors.

14.12.1.1 Good properties

In G': Results in power law degree distribution for degrees of nodes in both A and C. In G: results in power law distribution for A; yields densification of E,

shrinking/ stabilizing diameter.

14.12.1.2 Algorithmic benefits

G sparsifiable while mostly preserving distances from a set of relevant nodes.

14.13 Kronecker power graphs

14.13.1 Kronecker-multiplied graphs

(Leskovec et al). Take adjascency matreces A, B. Edge property: $A \otimes B$ is a graph with $Edge(X_{i,j}, X_{k,l})$ iff $(i,k) \in edges(A) \wedge (j,l) \in edges(B)$. So, like placing graph B in stead of every node of A, and then making extra edges. Could be disconnected if no self-loops. Defines Kronecker power graph $G_k = G^{(k)}$: G_1 assumed to have self loops for each node.
Can also use a stochastic adjascency matrix for G_1.

14.13.2 Properties

From Edge property: Degree distribution of G_k is kth Kronecker power of $(d_1..d_n)$: so multinomial. EW distribution is $(\lambda_1..\lambda_n)$. Components of each eigenvector follow a multinomial distribution. [**Find proof**]$E_k = E_1^k, N_k = N_1^k$: so follows densification power law.
From Edge property: If A, B have diameter at most d, $A \otimes B$ has diameter at most d. If $diam(G_1) = d$, $diam(G_k) = d$.
For stochastic Kronecker products: Degree distribution, scree plot shown to compare well with real world graphs. [**OP**]: Demonstrate the theoretical connection with the power law degree distribution.

14.13.3 Defects

Does it have community structure? Does it have a good clustering coefficient distribution? Presumed answer no.

14.14 Other generators

Redirection models: produce constant diameter, logarithmically increasing avg degree. Exponential random graphs

14.14.1 Variations

Slight tweaks to better imitate real graphs: eg: orphans.

14.15 Evolution of affiliation nw in social nw websites

Extends microscopic nw evolution model thus: At each time step, each awakened node v does this: pick number of groups to join, n from ke^{-kn} with mean $k^{-1} = rdeg(v)^g$: thus striving for exponential distribution of number of groups of nodes of given degree, polynomial growth of avg number of groups wrt degree; join n groups doing this: with prob t create a new group; else join existing group: with prob $p_v = \eta deg(v)$ join a group picked through a friend: η represents friend's influence; else join random group.
Yields same properties for social nw evolution
as microscopic evolution model; achieves power law group size distr; suitable fraction of singletons in groups (by tweaking t, η); exponential distr for num of affiliations for nodes of given degree.

15 Link prediction problem

Graph G = (V, E). Every edge has this label: t(e): The time during which interaction represented by e took place. Given $G[t_0, t_0']$ (training interval), output a list of edges not presented in it, but present in $G[t_1, t_1']$ (testing interval). Maybe want to output top n most likely-to-appear edges.

15.1 Motivation

Security. Improving organizational efficiency by going beyond official hierarchy. Inferring missing links from observed network.

15.2 Observations about link prediction

Small world problem: tenuous short links exist between otherwise unrelated nodes: noise wrt to the link prediction problem.
[2] observed that in their data-sets, new links are 3 or 4 times more often formed at distances ≥ 3. They compared performance of prediction methods at predicting new links between nodes at distances ≥ 3.

15.3 Using similarity measures between nodes

[2] experimented. Performance compared against a random predictor. Katz measure, and variants with added noise reduction performed well. [2] found that Katz, common neighbors in low rank approximation of G, Adamic/ Adar are similar in terms of common predictions. [**Incomplete**]

15.4 Reducing noise

Low rank approximation: Take adjacency matrix M; use SVD to get rank k approximation M_k of M wrt various norms. Can then use other similarity

measure based techniques. Usually, intermediate rank approximation performs best [2] .
Clustering: Remove tenuous edges.

15.5 Enhance similarity measure for nodes with similar neighbors

Adaptation of 'unseen bigrams' technique from language modelling: word pairs which occur in test corpus but not in training corpus. Take S_x^d: the top d nodes related to x under a similarity measure; get modified score(x,y) = $|\left\{z : z \in \Gamma(y) \cap S_x^d\right\}|$.

15.6 Model evolution of E

15.6.1 Using a linear model

Split E into 2 parts
thence get adj matreces A and B. Use various F like $F(A) = e^{aA}$; find parameters a of F to get $\|F(A) - (A + B)\|$. Take $A = U\Lambda U^*$: Λ decomposition; if $F(A) = UF(\Lambda)U^*$, this becomes: $\|F(\Lambda) - (A + B)\|$: this is a least squares curve-fitting problem like min $\sum(e^{a\lambda_i} - (A + B)_{i,i})^2$.

15.7 The experimental setup

15.7.1 The problem

Perhaps you have different snapshots of the graph for different times: test set will be the latest snapshot - previous snapshots. Perhaps you have a single snapshot of the graph: test set will be some random split of the graph into training and test edges. n edges to be predicted.
Often compare against a random predictor. The random predictor: What is the expected number of edges you get right if you pick n edges randomly?
To see understand behavior of a single set: Plot precision curve, recall curve.

15.7.1.1 The sparse case

If data is too sparse, there is not enough signal, and similarity measures such as Katz do worse than random. To compare various methods in this case: one picks many mn edges and then compare 'recall': the correct edges in the prediction set.

15.8 Open problems

[**OP**]:
Discover better performing link prediction algorithms. Make link prediction

algorithms based on distance metrics more efficient.
[**OP**]: Include more data in the social network (eg: paper titles, areas) and improve link prediction performance.
[**OP**]: Improve link prediction performance by giving greater weight to more recent collaborations.
[**OP**]: View training-period collaborations as samples drawn from a distribution on pairs of people; Try to use work on estimating distribution support in link prediction. [**OP**]: You only have +ve labelled examples; see if machine learning provides better learning heuristics in such cases.

16 Network cascades

16.1 Applications

Epidemology. Spread of ideas.

Part VI

Analysis of real world networks

17 Partitioning nodes

17.1 Number of partitions, k

See statistics ref for further info.

17.2 Minimum cut objectives

Maybe find $argmin_{(V_i)} cut((V_i))$: but $V_1 = V$ minimizes it. So, to balance the partitions, associate each vertex with a weight w(v), get W = diag(w(v)); define $w(V_i) = \sum_{v \in V_i} w(v)$; minimize $Q((V_i)) = \sum \frac{cut(V_i)}{w(V_i)}$.

17.2.1 Ratio, normalized cuts

If $w(v) = 1$, get ratio cut objective fn. So, ratio cut is trying to minimize the weight of cross-cut edges, averaged over the nodes.
If $w(v) = \sum E_{v,w}$, get normalized cut objective fn: $N((V_i)) = \sum \frac{cut(V_i)}{w(V_i)} = 2 - \sum \frac{within(V_i)}{w(V_i)}$: so minimizing N is same as maximizing wt of edges lying within each partition. Normalized cut is trying to minimize the weight of cross-cut edges, averaged over the nodes but normalized to allow high-degree vertex-sets to have more cross-cut edges.
These are discrete optimization problems: NP complete [**Find proof**]. Effective heuristics exist.

17.3 Modularity of partition C

(Newman) Amount by which number of edges in C exceed Chung's degree distribution preserving random graph model.

17.4 Spectral clustering

Core Idea: A $\|$ relaxation of normalized cut objective $\equiv$ finding a solution to the generalized eigenvector problem over the graph laplacian L, then partitioning points derived from the top k ev.

17.4.1 The objective

Represent cut by partition vector $p \in \{\pm 1\}^n$. Then Rayleigh quotient: $\frac{p^T L p}{p^T p} = n^{-1} 4cut(V_1, V_2)$: $p^T p = n; p^T L p = \sum_{e_{i,j} \in E} e_{i,j}(p_i - p_j)^2$.
Use partition vector q with $q_i = \sqrt{\frac{w(V_2)}{w(V_1)}}$ for $i \in V_1$; else $-\sqrt{\frac{w(V_1)}{w(V_2)}}$. Then $q^T W 1 = \sqrt{\frac{w(V_2)}{w(V_1)}} w(V_1) - \sqrt{\frac{w(V_1)}{w(V_2)}} w(V_2) = 0$; $q^T W q = w(V)$.
As $q = \frac{w(V_1)+w(V_2)}{2\sqrt{w(V_1)w(V_2)}} p + \frac{-w(V_1)+w(V_2)}{2\sqrt{w(V_1)w(V_2)}} e$; so $q^T L q = \frac{[w(V_1)+w(V_2)]^2}{4w(V_1)w(V_2)} p^T L p$. As $p^T L p = cut(V_1, V_2)$; $\frac{q^T L q}{q^T W q} = Q(V_1, V_2)$.
Minimizing this subject to suitable q's is still NP complete; so use real relaxation. So, solve $\min_q \frac{q^T L q}{q^T W q}$ subject to $q^T W e = 0$, deduce partition vector from solution. Opt problem solved when q is ev corresponding to λ_2 for generalized ev problem: $Lz = \lambda W z$. [**Find proof**]
To find k custers, do this recursively, or pick $m \geq \log k$ ev and partition points formed by putting various components of these ev together, maybe using k-means.
Requires $O(n^2)$ time: to find ev; $O(kn^2)$ to find k clusters among graph nodes.

17.4.2 Coclustering nodes in bipartite graph G=(A, B, E)

Core Idea:
|| (Dhillon). Finding the second ev of L ≡: take $|A| * |B|$ adjacency matrix M, and find its 2nd left and right sv: as M is smaller than L, ye save on computation.

17.5 Graclus

Does kernel-k-means among nodes. Equivalent to solving a relaxation of the normalized cuts objective.

18 Identifying dense subgraphs : relevant clusters

18.1 (a, b) cluster C

(Tarjan etal.) E(v, C): edges between v and C. External sparsity vs internal density of edges: $\forall v \in C : |E(v, C)| \geq b|C|$, $\forall u \notin C : |E(u, C)| \leq a|C|$. Cliques are (a, 1) clusters.
Bound on cluster overlap known. For certain (a, b) it is possible for one cluster to be contained in another, but not if size of largest : smallest cluster size $\leq \frac{1-a}{1-b}$. $b > 1/2$ ensures cluster connectedness: condition may be too strong in practice.
r-champion v of C: v has $\leq r|C|$ neighbors outside C. Gives a stronger argument for C being a good cluster.
Good clusters have small r and a, large b.

18.1.1 Finding (a, b) clusters

If there is a large gap between a/2 and $b > 1/2+(r+a)/2 > 1/2$, then there are $\leq n$ (a, b) clusters with r-champions of size s, can find them deterministically in $O(mn^{1.2} + n^{2+o(1)})$. Experimentally shown to find 90% of maximal cliques size 5 or higher.
Alg: Take $t(c) = \Gamma(c) \cup \Gamma(\Gamma(c))$. For each $c \in V$: start with $C = \phi$; for each $v \in t(c)$: add v to C if $|\Gamma(v) \cap \Gamma(c)| \geq (2b-1)s$ [1]; if C is an (a, b) cluster, output it.
[**OP**]: extend to weighted graphs.

19 Network of computers

Eg: Internet.

[1] Reminiscent of most new edges closing triangles in graph evolution

19.1 Important matrices

Matrices showing delay, traffic etc..

19.1.1 Matrix completion problems

Often these have missing values. So, use matrix completion approaches.

19.1.1.1 Peculiarities

Maybe try to embed network: but often triangle inequality and symetry doesn't hold.
Missing values tend to be highly structured. But many matrix completion methods assume that missing values tend to be uniformly distributed.

Part VII

Mapping Graphs and pts in D dim Euclidean space

20 Graphs from pts in Euclidean space

See statistics ref.

21 Embedding complete graph with distance weights

Want the minimum dimensional embedding.

21.1 Formulation as matrix factorization

(Shconberg). Take weighted adj matrix W. $W_{i,j} = \|x_i - x_j\|^2 = x_i^T x_i + x_j^T x_j - 2x_i^T x_j$. Take $X = (x_i)$; Gram matrix $G = X^T X$ normalized to have $g_{i,i} = 1$; then $W = 2.11^T - 2G$. Then solve for G; then solve $G = X^T X$ for X with min number of columns.

21.2 Energy minimization techniques

See section on incomplete graphs with similarity measures for intro to notation. Use energy fn: $U = \sum_{(i,j)}(f(d_{i,j}) - g(d_{i,j}))$. Minimize over R^D.

21.2.1 Clustering postulate and constraint

Require $d_{min} = w(1/C)^\lambda$ in attempting to place clusters c_1, c_2 with coupling C at distance w; so we want $\frac{1}{C} = (\frac{d}{w})^{1/\lambda}$. Then, $U = f(d) - \frac{1}{C}g(d)$. Set derivative to 0, get constraint: $f'(d) = (\frac{d}{w})^{1/\lambda} g'(d)$.

21.2.2 Box Cox family of energy fn

$U = \sum_{i,j} BC_{\mu+\frac{1}{\lambda}}(d_{i,j}) + D_{i,j}^{1/\lambda} BC_\mu(d)$.
Encompasses energy fn used in multidimensional scaling.

21.3 Applications

If nodes represent sample-points, can infer the dimensionality of the sample space. Then can measure distance distance between arbitrary sample points. Eg: Maybe can sample some hyenas, observe their interactions, conclude that Hyenas are 5 dimensional.

22 Embedding Incomplete Graph G

Can view this as matrix completion problem; perhaps with additional constraints enforcing symmetry and traingle inequality for the distance metric.

22.1 Energy minimization techniques for G with similarity wts

Put attractive forces f() between verteces connected by edges, and repulsive forces g() between all vertex pairs, define energy: $U = \sum_{(i,j)\in E} f(d_{i,j}) - \sum_{i,j} g(d_{i,j})$ then find minimum energy configuration in R^D.

22.1.1 Clustering postulate and constraint

Constrain range of choices for these forces with clustering postulates.
Take graph with 2 clusters c_1, c_2 with coupling $C = \frac{E(c_1,c_2)}{E(c_1)E(c_2)}$. Require $d_{min} = \frac{1}{C}^\lambda$ be minimum energy configuration; λ is clustering power.
Setting $U'(c_1, c_2) \approx f'(d_{min}) - (1/C)g'(d_{min}) = 0$, yields the clustering constraint: $f'(d) = d^{1/\lambda} g'(d)$.

22.1.2 Box Cox family of energy functions

Want to allow cases where $f(d) = d, g(d) = \log d$; so use Box-Cox transformations for f, g: $d > 0 : BC_p(d) = \frac{d^p - 1}{p}$ if $p \neq 0$, else $\log d$; with $BC'_p(d) = d^{p-1}$. So, use $g(d) = BC_\mu(d)$, get $f(d) = BC_{\mu+\lambda^{-1}}$.

22.2 Eigenmap

Take laplacian L of the graph G, and take its ev corresponding to bottom few ew, which are functions over V which are smooth over E.

22.3 G with distance wts D

This can be reduced to a complete graph: calculate distance for all pairs by finding shortest paths.

Bibliography

[1] Jure Leskovec. *Dynamics of large networks*. PhD thesis, CMU, 200?

[2] David Liben-Nowell and Jon Kleinberg. The link-prediction problem for social networks. *Journal of the American Society for Information Science and Technology*, 58(7):1019–1031, May 2007.

[3] Elena Zheleva, Hossam Sharara, and Lise Getoor. Co-evolution of social and affiliation networks. pages 1007–1016, 2009.