Graph Analytics

Data Networks and Graphs

DATA NETWORK: set of entities together with a number of pairwise relations among them. Each entity/relation can be provided with additional information (attributes)

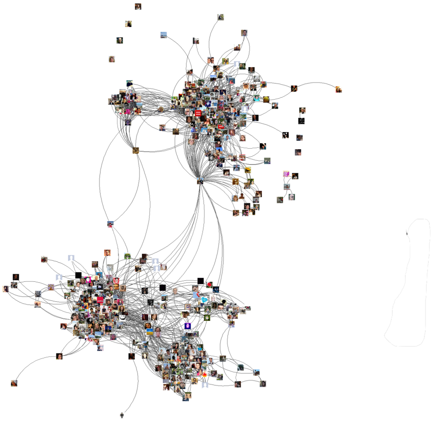
A data network can be conveniently represented as a GRAPH $G = (V, E)$

- V: (nodes) represents the entities, possibly with attributes
- $E \subseteq V \times V$: (edges) represents the pairwise relations between entities, possibly with attributes.

The graph is undirected (resp. directed) if edges are unordered (resp. ordered) pairs, and is weighted if each edge is associated with a numerical attribute representing a weight.

Examples: Social networks

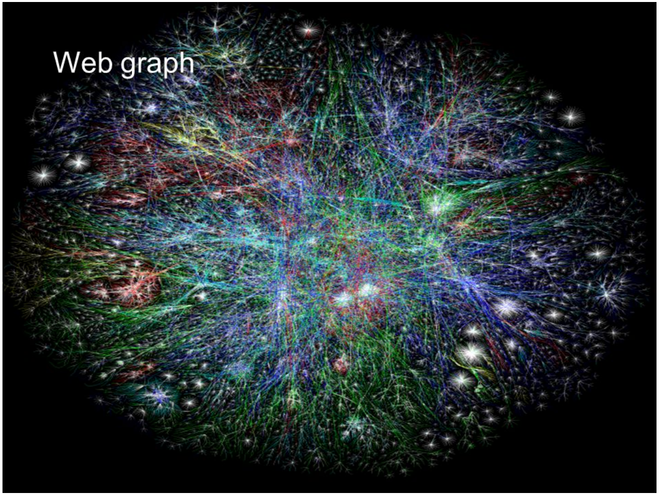
Facebook graph (about 2 billion users)



Also: Google+, LinkedIn, Twitter ..

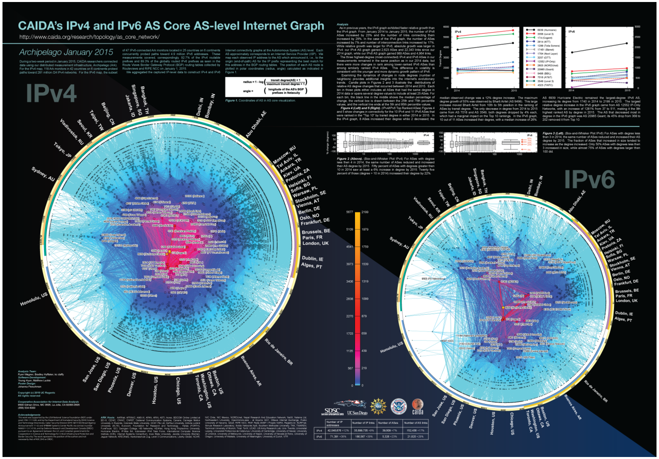
Examples: World Wide Web

Web graph (over 1 billion sites)



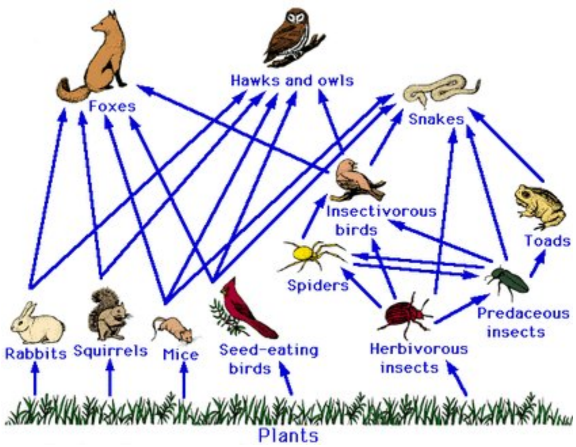
Examples: Communication networks

AS-level Internet graph (40k-50k nodes)



Examples: Biological networks

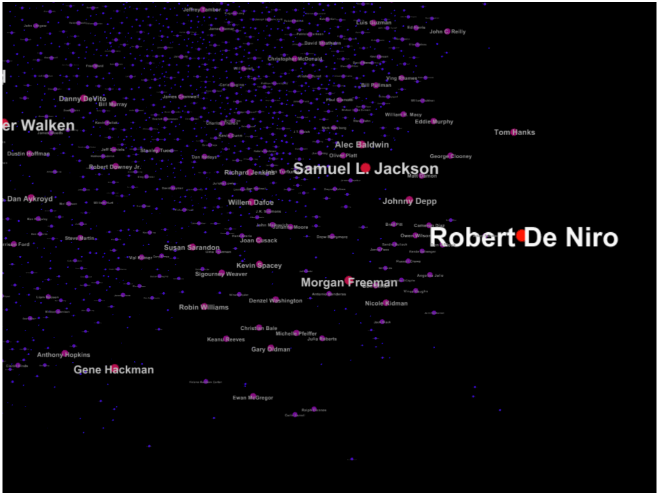
Food chain network



Also: Protein-Protein Interaction networks, biological neural networks, ..

Examples: Collaboration networks (actors)

Actors collaboration network



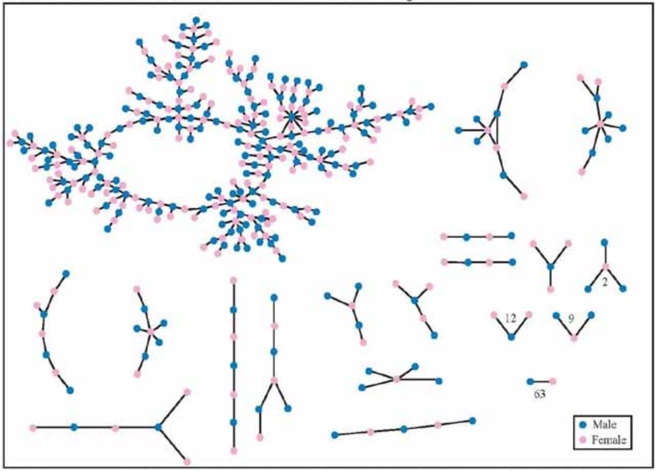
Also: co-authorship network (e.g., DBLP), citation networks

Examples: Road networks

Examples: Romantic networks

Romantic-sexual relationships at a US High School (832 students)

The Structure of Romantic and Sexual Relations at "Jefferson High School"



Each circle represents a student and lines connecting students represent romantic relations occuring within the 6 months preceding the interview. Numbers under the figure count the number of times that pattern was observed (i.e. we found 63 pairs unconnected to anyone else).

Examples: etc.

- Google+, LinkedIn, Twitter networks
- Citation networks, Co-autorship networks
- P2P networks
- Power grid networks
- Telephone networks
- Airline, railways networks
- Protein-Protein Interaction networks
- Biological neural networks
- etc.

Graph analytics: objectives

1. Connectivity analysis: e.g., s-t reachability, connected components, bridges, spanning trees/forests
2. Distance analysis: e.g., distances between nodes, diameter, radius
3. Centrality analysis: e.g., influential nodes, centralities
4. Community analysis: e.g., community detection (clustering), clustering coefficient, triangle count
5. Pattern mining: e.g., frequent subgraphs (motifs), subgraph enumeration

Remark: most problems are easily solved for small graphs (say up to 100k nodes) but become increasingly challenging, from a computational point of view, for larger graphs

Graph analysitcs: applications (some examples)

- Route/traffic optimization for logistics, distribution chains, smart cities
- Discovery of weaknesses in utility power grids ior transportation networs
- Discovery of influencers or communities in a social network
- Fraud, money laundering detection in Financial Activity Networks (FANs)
- Discovery of mutation patterns associated with diseases in gene interaction networks

Fundamental Primitives

Graph $G = (V, E; w)$: n nodes (set V), m edges (E), weights $w : E \to \Re$

- Breadth-First Search (BFS), Depth-First Search (DFS): systematic explorations of all nodes and edges of G.

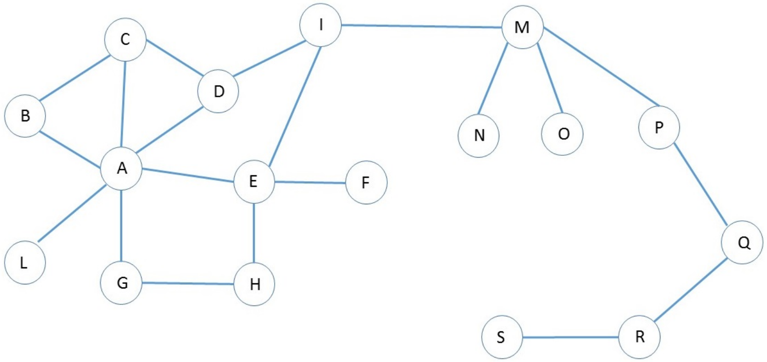
  Sequential complexity: $O(n + m)$

- Single Source Shortest Paths (SSSP): shortest paths from a source node $s \in V$ to all other nodes

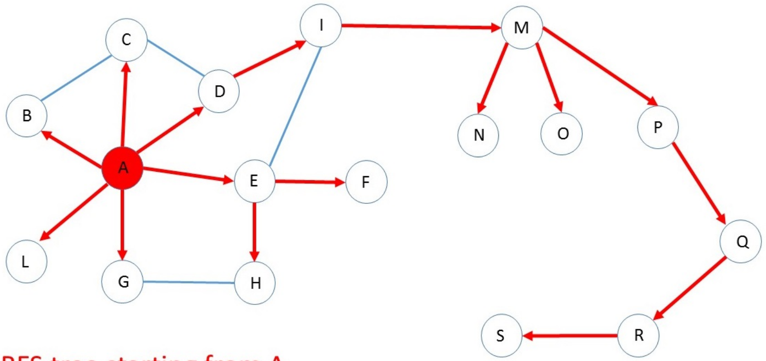
  Sequential complexity:
  - Unweighted (i.e., unit weights): $O(n + m)$ (via BFS)
  - Nonnegative weights: $O(n \log n + m)$ (Dijkstra)
  - Arbitrary weights (no negative cycles): $O(nm)$ (Bellman-Ford)

N.B. In what follows, we focus on undirected graphs. Adaptations to directed graphs are sometimes trivial, but other times are tricky!

Example: graph



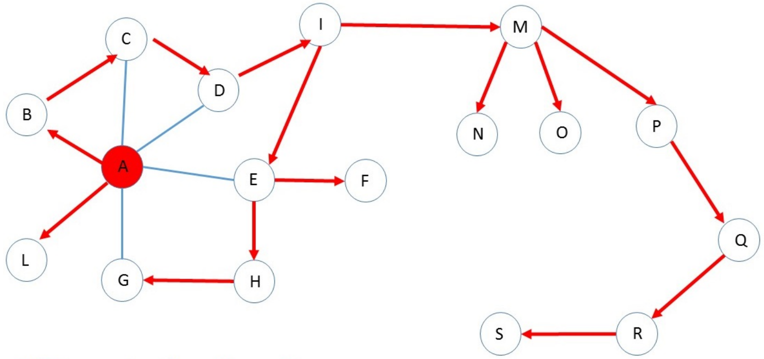
Example: BFS



BFS tree starting from A

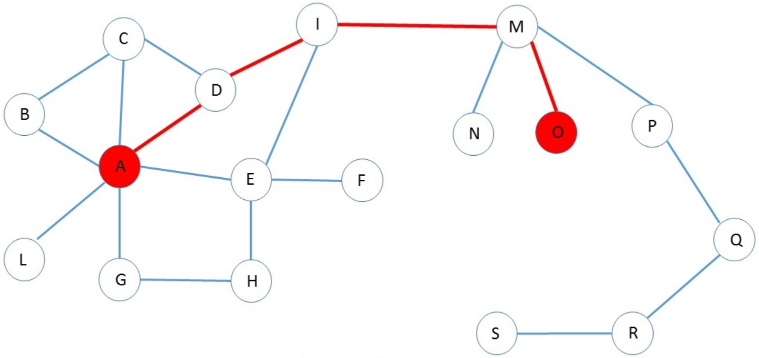
(arrows indicate direction of discovery)

Example: DFS



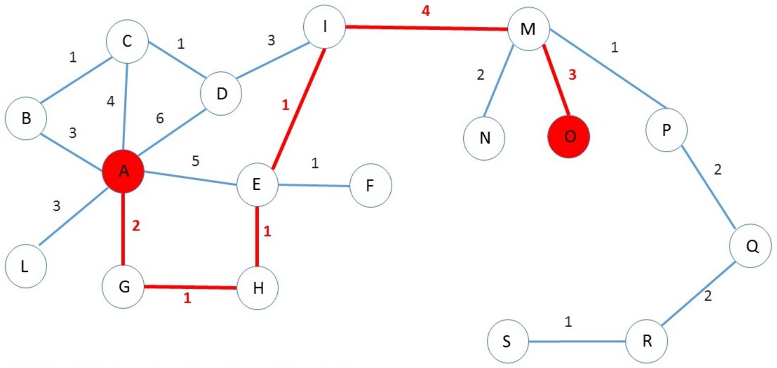
DFS tree starting from A
(arrows indicate direction of discovery)

Example: shortest path (unweighted)



Unweighted shortest path: A-O

Example: shortest path (weighted)



Weighted shortest path: A-O

(weight=12)

Fundamental Primitives (cont'd)

- Connected Components: maximal connected subgraphs of G

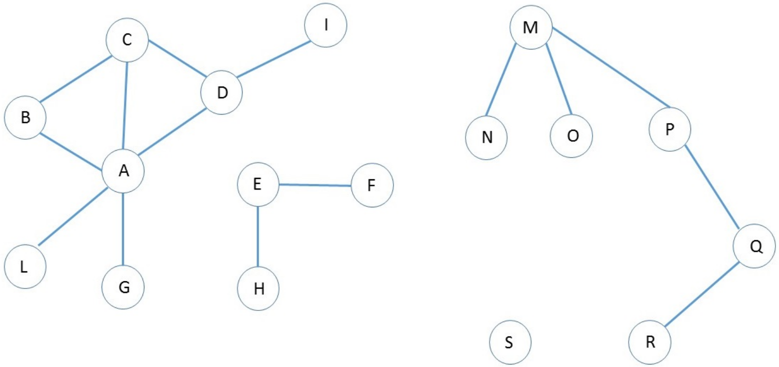
  Sequential complexity: $O(n + m)$ (via BFS or DFS)

- Minimum Spanning Tree (MST) or Forest: spanning tree for each connected component of minimum total weight

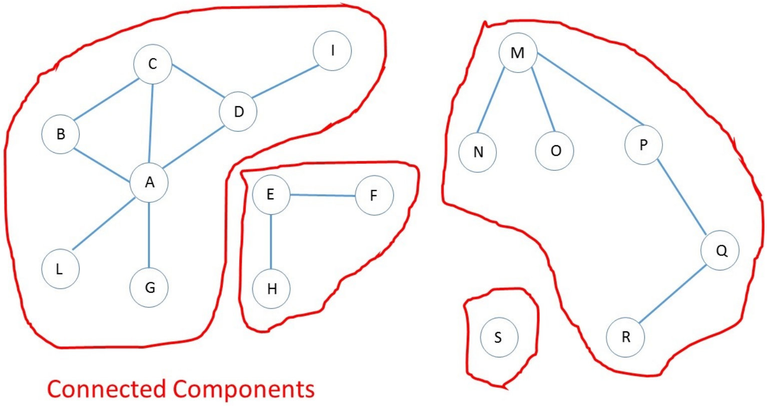
  Sequential complexity: $O(n \log n + m)$ (Prim's algorithm)

Remark: Faster MST algorithms for sparse graphs ($n \simeq m$) exist

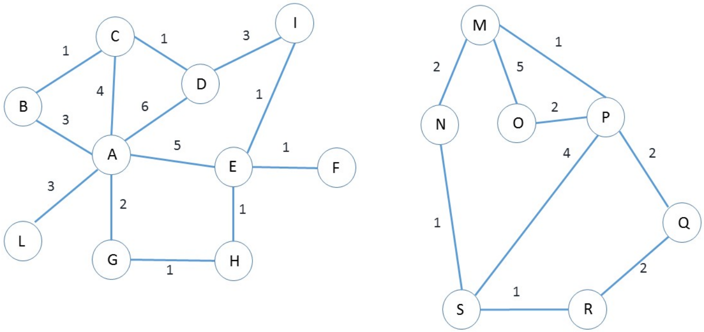
Example: Connected Components



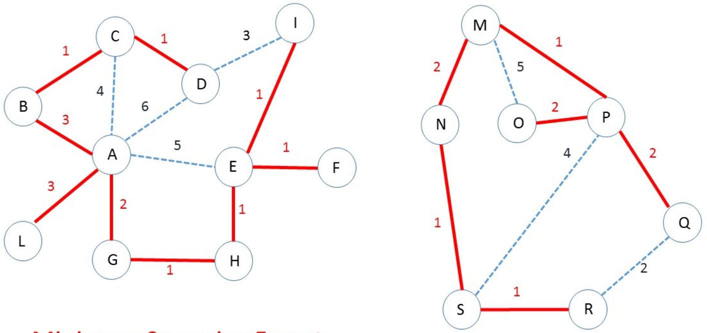
Example: Connected Components



Example: Minimum Spanning Forest



Example: Minimum Spanning Forest



Minimum Spanning Forest
(total weight = 23)

Challenges

For typical graph analytic tasks:

- Exact solutions require superlinear number of operations
- Parallel strategies incur high communication costs (e.g., several rounds in MapReduce)
- Sampling approaches are hard to devise
- Sparse inputs are often tougher

Filtering

Filtering: reduce the size of an input graph by eliminating (*filtering out*) or contracting some edges until the graph is small enough to be processed efficiently (e.g., by a single reducer)

We demonstrate this technique for computing the MST and Connected Components of a large dense graph.

Input Weighted Graph $G = (V, E; w)$, with $|V| = n$ and $|E| = m = n^{1+c}$ for some constant $c > 0$

Output Minimum spanning forest of G, that is, MST of each connected component of G.

Remark: *From this output, one can immediately derive the connected components of G or assess whether G is connected*

MR algorithm for minimum spanning forest ([L+11])

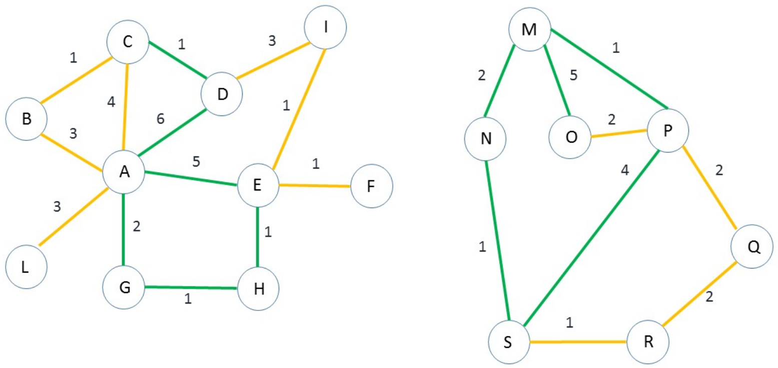
- **Round 1:** Partition E at random into $\ell = n^{c/2}$ subsets $E_1, E_2, \ldots, E_\ell$, and compute the minimum spanning forest of the graph induced by each E_i separately. Let $F_i \subset E_i$ be the set of edges of the i-th forest, $1 \leq i \leq \ell$
- **Round 2:** Compute (with a single reducer) and return the minimum spanning forest of $G' = (V, F_1 \cup F_2 \cup \cdots \cup F_\ell)$, subgraph of G.

Observation: The first round filters out some edges, leaving at most $n-1$ edges in each F_i (*recall from DA1 that a forest over n nodes has at most $n-1$ edges!*).

Assuming that for every $1 \leq i \leq \ell$, $|E_i| = O(m/\ell) = O\left(n^{1+c/2}\right)$, which is true with high probability (Chernoff+union bound), the local/aggregate space requirements of the algorithm are

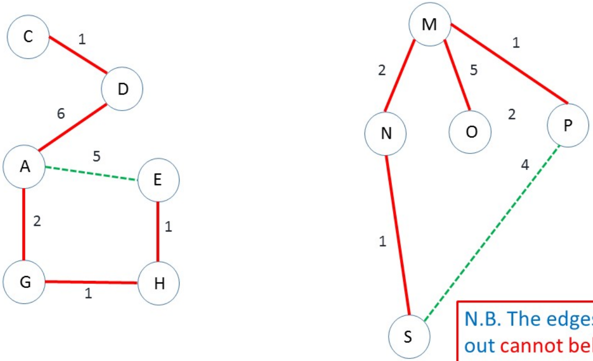
- $M_L = \Theta\left(n^{1+c/2}\right) = o(m)$ (i.e., sublinear in the input size)
- $M_A = \Theta(m)$ (i.e., linear in the input size)

Example



Partition into 2 subsets: E_1 E_2

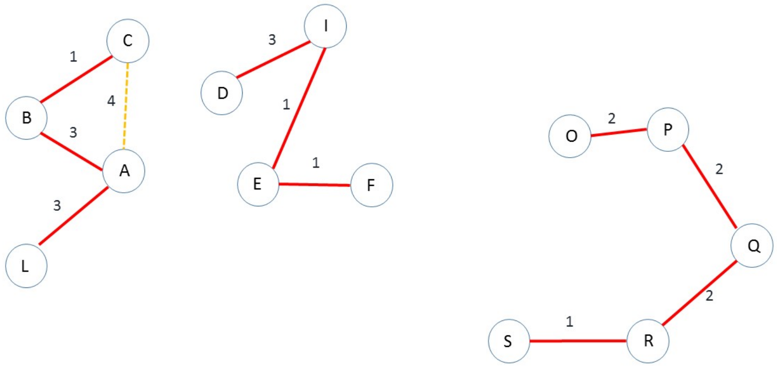
Example (cont'd)



Minimum spanning forest for E_1

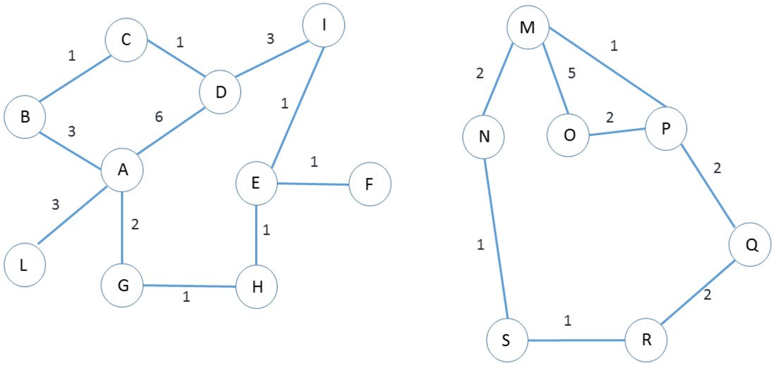
N.B. The edges filtered out cannot belong to any minimum spanning forest of the original graphs!

Example (cont'd)



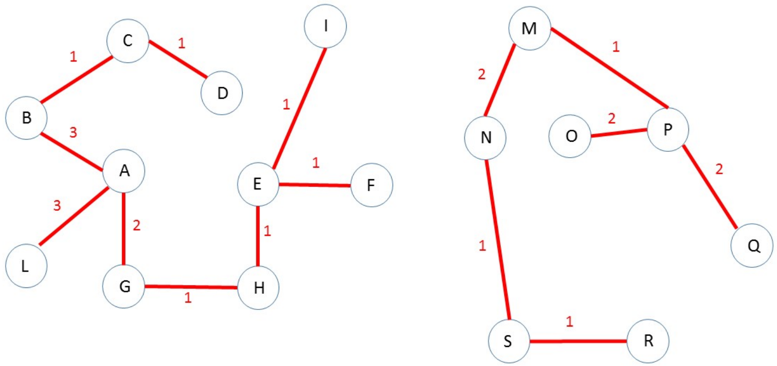
Minimum spanning forest for E_2

Example (cont'd)



Residual graph G' after filtering

Example (cont'd)



Minimum spanning forest

Correctness

Theorem

The MR algorithm for minimum spanning forest is correct.

Proof

It is sufficient to show that there exists a minimum spanning forest that uses none of the edges filtered out in the first round. Let H be an arbitrary minimum spanning forest for G. For each edge (u, v) belonging to H with $(u, v) \in E_i - F_i$, for some i

- u and v must belong to the same tree T of the forest defined by F_i, otherwise there would be two trees for the same connected component
- Adding (u, v) to T creates a cycle C where the weight of (u, v) is at least as large as the weight of any other edge in C, otherwise T would not have minimum weight (*cycle property*)

Correctness (cont'd)

Proof of theorem (cont'd).

- Remove (u, v) from H, thus splitting some tree $T' \subseteq H$ into 2 subtrees $T'(1)$ and $T'(2)$, and replace (u, v) in H by one of the edges of C to join again the subtrees (note that an edge in C which joins $T'(1)$ and $T'(2)$ must exist!). The weight of the new spanning forest we obtain is not larger than the weight of H.

After processing each edge (u, v) of H not belonging to the F_i's as explained above, we get a new minimum spanning forest using only edges of the F_i's $\square$

Trading rounds for local space

The MR algorithm for minimum spanning forest can be generalized to work with $O\left(n^{1+\epsilon}\right)$ local space, for any $0 < \epsilon < c$ (recall that m^{1+c}), using the following strategy. Let $M = n^{1+\epsilon}$ and $m_0 = m$.

- **Round 1:** Partition the edges at random into $\ell_0 = m_0/M$ subsets and compute a minimum spanning forest separately for each subset. Let m_1 be the number of residual edges (i.e., edges that belong to computed forests)
- **Round 2:** Partition the m_1 residual edges at random into $\ell_1 = m_1/M$ subsets and compute a minimum spanning forest separately for each subset. Let m_2 be the number of residual edges.
- $\ldots$ Continue until, at the end of **Round** i, we have $m_i \leq M$. Then, in the next round (**Round** $i+1$) compute the final minimum spanning forest on the m_i residual edges.

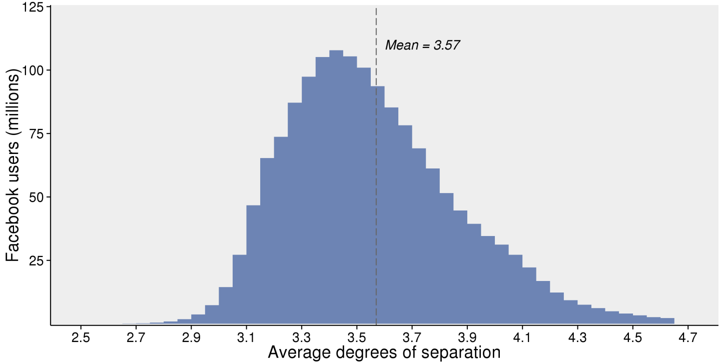
Exercise

Consider the above strategy for computing the minimum spanning forest in MapReduce

- Give an upper bound to the number of rounds as a function of c and ϵ.
- Consider a Round j, with $1 \leq j \leq i$, and one specific subset S of the random partition of the m_{j-1} residual edges, performed at the beginning of the round. Determine an upper bound to the probability that $|S| \geq 6M$ (use the Chernoff bound).
- Using the above points, show that the probability that the implementation requires local space $< 6M$ tends to 1 as n goes to infinity.

Distances (or degrees of separation)

Average degrees of separation (i.e., "distance"-1) between nodes in the facebook graph, as of february 2016: nodes are user profiles (about 1.6 billion at that time); undirected edges are friendship relationships [B+16].



Remark: *The graph is not connected and the average is taken over all connected pairs*

A bit of history

- 1929: In a short story (*Chains*) Krigyes Karinthy conjectures that any two people are connected by a chain of at most 5 intermediaries. Hence, the theory of 6 degrees of separation (here, erroneously regarded as "distance")
- 1967: Stanley Milgram does an experimental validation of the theory: asks a group of people from Midwest (USA) to send a parcel to an unkown recipient in Massachusetts , through friends, friends of friends, etc. It discovers that the average number of intermediaries is between 4.4 and 5.7
- In 2011, Boldi et al. [B+11] analyze the FB graph (at that time with about 721 million nodes and 69 billion edges) and discover
  - Average degrees of separation: 3.74. For 92% of the connected pairs it is ≤ 4
  - Connectivity: the graph is not connected but has a giant connected component with $> 97\%$ of the nodes and diameter 41.

They made it to the New York Times (nov. 22, 2011)!

Computational issues

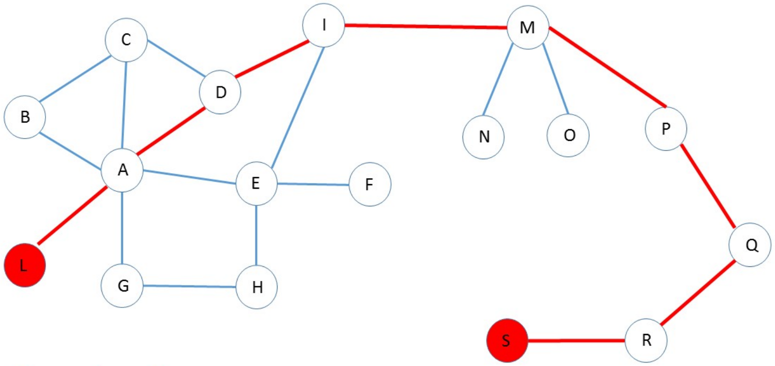
Undirected graph $G = (V, E)$. $\forall u, v \in V$, let $\text{dist}(u, v)$ be the length of the shortest path in G between u and v (i.e., minimum number of edges to reach v from u and viceversa). Set $\text{dist}(u, v) = \infty$ if u and v do not belong to the same connected component.

Define

$$\text{ADS}(G) = \frac{\sum_{u,v \in V \,:\, \text{dist}(u,v)<\infty} \text{dist}(u, v)}{|\{u, v \in V \;:\; \text{dist}(u, v) < \infty\}|} - 1$$

$$\text{Diameter}(G) = \max\{\text{dist}(u, v) \;:\; u, v \in V \wedge \text{dist}(u, v) < \infty\},$$

where ADS stands for Average Degrees of Separation.



Diameter = 8

Computational issues (cont'd)

How easy is it to compute ADS and Diameter? Conceptually easy. Recall that the BFS from a node v provides the distance between v and all other nodes reachable from v.

Thus:

Run BFS from each node

Requires, $\Theta(n \cdot m)$ operations, which is unfeasible for graphs with billion nodes/edges!

$\Rightarrow$ resort to approximate solutions

Neighborhood function

Consider an undirected, unweighted graph $G = (V, E)$.

Definition (Neighborhood function)

The neighborhood function $N_G(t)$ of G returns for each integer $t \geq 1$ the number of pairs of nodes $u \neq v \in V$ such that $\mathrm{dist}(u, v) \leq t$, where $\mathrm{dist}(\cdot, \cdot)$ denotes the shortest path distance.

Setting $N_G(0) = 0$, we have

$$\begin{aligned}
\text{ADS} &= \frac{\sum_{t \geq 1} t \cdot (N_G(t) - N_G(t-1))}{\sum_{t \geq 1} (N_G(t) - N_G(t-1))} - 1 \\
\text{Diameter} &= \min\{t \geq 1 \ : \ N_G(t) = N_G(t+1)\}
\end{aligned}$$

Exact Neighborhood Function

The following algorithm computes the exact neighborhood function for a graph $G = (V, E)$ with n nodes and m edges.

The algorithm is based on the observation that $\forall v \in V$ and $t \geq 1$,

$$\{z \in V \ : \ \text{dist}(v, z) \leq t + 1\} = \cup_{w \, : \, (v,w) \in E} \{z \in V \ : \ \text{dist}(w, z) \leq t\}$$

```
for each $v \in V$ do $C_v \leftarrow \{w : (v, w) \in E\}$ // direct neighbors of $v$
$N_G(0) \leftarrow 0$;  $N_G(1) \leftarrow \sum_{v \in V} |C_v|/2$;  $t \leftarrow 1$
repeat
    for each $v \in V$ do $C'_v \leftarrow \cup_{w \, : \, (v,w) \in E} C_w$
    for each $v \in V$ do $C_v \leftarrow C'_v$
    // at this point $C_v$ is the set of nodes at distance $\leq t + 1$ from $v$
    $N_G(t + 1) \leftarrow (1/2) \sum_{v \in V} |C_v|$ // each pair is counted twice in the sum
    $t \leftarrow t + 1$
until $(N_G(t) = N_G(t - 1))$
```

Approximate Neighborhood Function

The previous algorithm:

- Requires $\Theta(n)$ space at each node
- Performs d iterations, where d is the diameter of G,
- In every iteration computes, for each node, the union of sets of size up to $n-1$

$\Rightarrow$ not practical!

HyperANF: efficient implemention of the above strategy devised by [BRV11], which uses an approximate representation of each C_v (say $\tilde{C}_v$)

The main features of this representation are:

- Each $\tilde{C}_v$ requires only $O(\log\log n)$ bits!
- The approximate representation of $\cup_{w\,:\,(v,w)\in E} C_w$ is easily computed as a function of the $\tilde{C}_w$'s.
- An accurate estimate of $|C_v|$ is easily computed from $\tilde{C}_v$

$\Rightarrow$ $N_G(t)$ is accurtely estimated for each t. Moreover, the value $t-1$ after the last iteration provides a lower bound to the diameter of G.

Observations

- HyperANF can be employed to provide accurate estimates of the average degrees of separation and a lower bound to the diameter of the graph. However, for the diameter there are no theoretical guarantees on the accuracy of the estimate.
- While for high-diameter graphs (i.e., $d \in \Theta(n)$) the complexity of HyperANF is not better than the one of the trivial BFS-based algorithm, for low-diameter graphs (e.g., social networks) it runs rather fast
- HyperANF is very efficient on multicores with a lot of RAM

Approximating the diameter

Suppose $G = (V, E)$ is connected

- Run BFS from an arbitrary node $v \in V$
- Let $\Delta = \max\{\text{dist}(v, u) \ : \ u \in V\}$. Then

$$\Delta \leq \text{Diameter}(G) \leq 2\Delta$$

Thus, with just one BFS ($O(n + m)$ time) we can get an estimate of the diameter within a factor 2.

How hard is it to get a tighter estimate?

Theoretically, quadratic time is required to get an estimate within a factor less than 1.5. In practice, however, one can do better

Practical and accurate diameter estimation

Need two primitives. Let $v \in V$

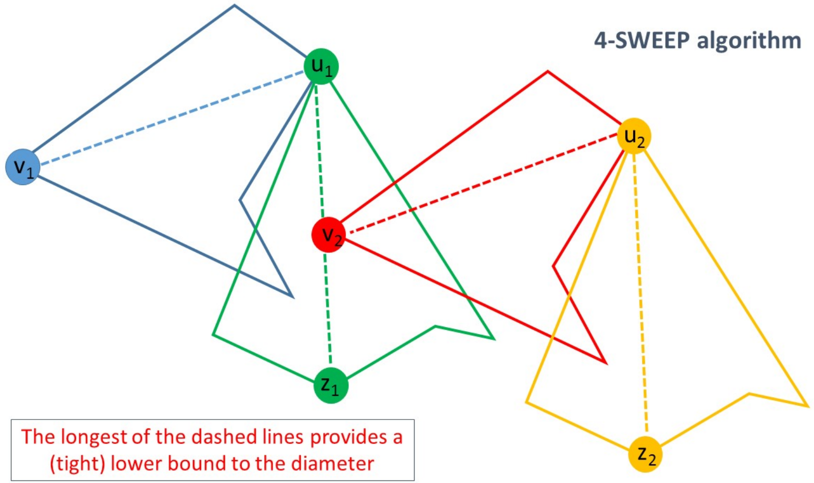
- max-dist(v): returns the node $u \in v$ with the largest shortest-path distance from v
- middle(v, u): returns a node w in the middle of a shortest path between v and u

Observation: both primitives require $O(n + m)$ time (BFS from v)

4-SWEEP Algorithm

```
$v_1 \leftarrow$ arbitrary node in $V$
$u_1 \leftarrow$ max-dist($v_1$)     /*1st sweep*/
$z_1 \leftarrow$ max-dist($u_1$)     /*2nd sweep*/
$v_2 \leftarrow$ middle($u_1, z_1$)
$u_2 \leftarrow$ max-dist($v_2$)     /*3rd sweep*/
$z_2 \leftarrow$ max-dist($u_2$)     /*4th sweep*/
return $\Delta = \max\{\text{dist}(v_1, u_1), \text{dist}(u_1, z_1), \text{dist}(v_2, u_2), \text{dist}(u_2, z_2)\}$
```

Example



The longest of the dashed lines provides a (tight) lower bound to the diameter

Observations

- In theory we can only claim that the value Δ returned by the 4-SWEEP algorithm is such that

$$\Delta \leq \text{Diameter}(G) \leq 2\Delta,$$

however, in practice Δ is very close to the actual diameter!

- In general, the diameter can be approximated with increasing accuracy (up to exact estimation) using a sufficiently large number of BFSes. One of the exercises explains why.
- For weighted graphs, the BFS-based diameter estimation algorithms can be employed by substituting each BFS with a SSSP computation.

Centrality

From [BV14]:

Real-world complex networks [...] typically generated directly or indirectly by human activity and interaction (and, therefore, hereafter dubbed "social"), appear in a large variety of contexts and often exhibit a surprisingly similar structure. One of the most important notions that researchers have been trying to capture in such networks is node centrality: ideally, every node (often representing an individual) has some degree of influence or importance within the social domain under consideration, and one expects such importance to surface in the structure of the social network; centrality is a quantitative measure that aims at revealing the importance of a node.

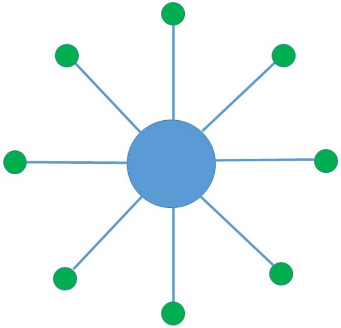
Spectrum of uses

Centrality has been used for decades as a tool in the analysis of (social) networks

- In the late 40s, Alex Bavelas, American psychologist at MIT defined a notiuon of centrality for revealing the impact of communciation patterns in a group's performance
- Since then, centrality has also been used to, understand political integration in Indian social life, to study communication paths for urban development, to explore efficient design of organizations, to explain the wealth of the Medici family w.r.t. marriages and finantial transactions in the 15th century (see links in [BV14])
- Centrality is also used for ranking purposes (e.g., in the context of web search)

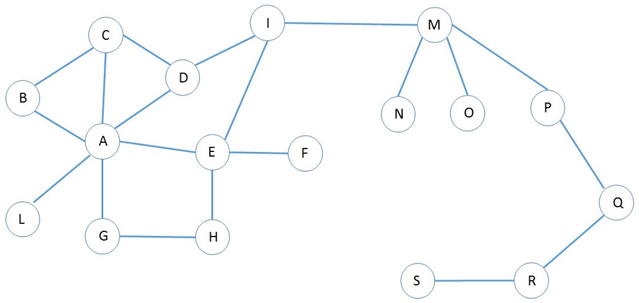
Measures of centralities

- There are many alternative measures of centrality of a node in a graph. The excellent survey [BV14] distinguishes among different types of measures: geometric, which are distance-based (e.g., *closeness*); and path-based (e.g., *betweennes*); spectral (e.g., *page rank*);
- While quite different from one another, most known definitions of centrality stem from the natural idea that a node at a center of a star is more central than the periphery



Degree centrality

Given a graph $G = (V, E)$ the simplest measure of centrality of each node $v \in V$ is its degree (if G is undirected) or in-degree (if G is directed)



Ranking by decreasing degree centrality:

A(6), E(4), M(4), C(3), D(3), I(3), B(2), G(2), H(2), P(2), Q(2), R(2), F(1), L(1), N(1), O(1), S(1)

Closeness centrality

The notion of closeness centrality is similar to the one introduced by Bavelas and is among the oldest and most popular ones.

(Undirected) graph $G = (V, E)$ with $|V| = n$ and $|E| = m$.

For each $v \in V$

$$\begin{aligned} \text{farness:} \quad f(v) &= \frac{\sum_{u \in V:\, u \neq v} \operatorname{dist}(u, v)}{n-1} \\ \text{centrality:} \quad c(v) &= \frac{1}{f(v)}, \end{aligned}$$

where $\operatorname{dist}(u, v)$ denotes again the shortest path length.

Remark: *If G is not connected, $f(v) = \infty$ and $c(v) = 0$ for each node, hence the measure becomes uninteresting*

Closeness centrality (cont'd)

The following generalization (a.k.a. Lin's index) makes closeness centrality meaningful also when G is not connected:

For each $v \in V$ let $n(v) = |\{u \in V \ : \ u \neq v \wedge \text{dist}(u,v) < \infty\}|$.

Then

$$\begin{aligned} \text{farness:} \quad f(v) &= \frac{\sum_{u \in V \,:\, u \neq v \wedge \text{dist}(u,v) < \infty} \text{dist}(u,v)}{n(v)} \cdot \frac{n-1}{n(v)} \\ \text{centrality:} \quad c(v) &= \frac{1}{f(v)}. \end{aligned}$$

Observations:

- *If G is connected, $f(v)$ and $c(v)$ are as before (since $n(v) = n - 1$ for each v)*
- *High $c(v) \Rightarrow v$ is reached by many nodes through short paths*

Closeness centrality (cont'd)

For simplicity, suppose that $G = (V, E)$ is connected

- Computing the closeness centrality $c(v)$ for a given node v can be done by running a BFS (or a SSSP in case G is weighted) from v.

  $\Rightarrow$ computing the closeness centralities of *all nodes* (e.g., to rank them) requires n BFSes!

- Eppstein and Wang's method [EW04] to efficiently estimate all centralities: pick $k < n$ random pivots $v_1, v_2, \ldots, v_k \in V$ and run BFS from each of them. Then, for each $v \in V$ compute the approximate closeness centrality

$$\tilde{c}(v) = \begin{cases} c(v) & \text{if } v = v_i, \quad 1 \le i \le k \\ \frac{k(n-1)}{n \sum_{i=1}^{k} \text{dist}(v_i, v)} & \text{otherwise} \end{cases}$$

Closeness centrality (cont'd)

In [EW04], Eppstein and Wang show that:

1. $1/\tilde{c}(v)$ is an unbiased estimator of $1/c(v)$, that is, its expectation (over all possible selections of pivots) is $1/c(v)$
2. For any $\epsilon \in (0,1)$, using $k = \Omega\left(\log n/\epsilon^2\right)$ pivots ensures that with high probability, for each node $v \in V$ the value $1/\tilde{c}(v)$ approximates $1/c(v)$ within an additive factor at most $\epsilon D(G)$, where $D(G)$ is the diameter of the graph.

Remark: Several recent works have developed more refined methods to get tighter estimates of the closeness centralities, or to identify the nodes with the top-K centralities.

MapReduce considerations

- It is difficult to devise MapReduce algorithms that perform graph analytics on large, sparse graphs with low round complexity.
- Straightforward implementations of HyperANF or of BFS-based algorithms require $\Omega\,(\text{diameter})$ rounds if sublinear local space (and linear aggregate space) is desired.
- For diameter estimation, there exist clustering-based strategies that attain a better round complexity at the expense of a (theoretically) worse accuracy.

Other centrality measures: harmonic centrality

Let $G = (V, E)$ be an undirected graph with n nodes.

Harmonic centrality: for a node $v \in V$, its harmonic centrality is defined as:

$$h(v) = \frac{1}{n-1} \sum_{u \in V:\ u \neq v} \frac{1}{\mathrm{dist}(u, v)},$$

where we assume that $1/\infty = 0$.

Comment: it is a distance-based measure similar in spirit to the closeness centrality but well defined also for disconnected graphs.

Other centrality measures: betweenness centrality

Betweenness centrality: for any three nodes u, v, w, let σ_{uw} be the number of distinct shortest paths from u to w, and let $\sigma_{uw}(v)$ be the number of such paths that pass through v. The betweenness centrality of a node $v \in V$ is defined as:

$$b(v) = \sum_{u,w \in V:\ u \neq v \neq w} \frac{\sigma_{uw}(v)}{\sigma_{uw}}$$

Comment: the intuition behind this measure (which belongs to the class of path-based centrality measures, is that a node that belongs to many shortest paths is an important junction for the graph and should therefore be considered *more central*.

Other centrality measures: page rank

Consider an *abstract surfer* than navigates a directed graph G starting from an arbitrary node. Suppose that at some point the surfer is positioned at node $v \in V$. If v has no outgoing edges, then the surfer will move to a random node in V, uniformly chosen. Otherwise, the surfer will move to a neighbor along any of the $d_v > 0$ outgoing edges, with probability $\alpha \cdot 1/n_v$, or to an arbitrary node in V, uniformly chosen, with probability $1 - \alpha$, for some $\alpha \in (0, 1)$.

Page rank: the page rank of a node $v \in V$ is the probability, at steady state, that the surfer is in node v.

Comment: This measure was originally proposed to rank web pages and captures the idea that an important (i.e., central) page should attract more the interest of a generic surfer. The factor α is called damping factor and represents the chance that the web surfer stops following hyperlinks and decides to restart navigation from a random node.

Exercises

Exercise

Let $G = (V, E)$ be a connected, undirected graph with n nodes. Suppose that a BFS is executed from each of $k > 1$ distinct nodes $v_1, v_2, \ldots, v_k \in V$ and that the following two values are computed:

$$\begin{aligned} R &= \max_{u \in V} \min_{1 \le i \le k} \mathrm{dist}(u, v_i) \\ \Delta &= \max_{1 \le i, j \le k} \mathrm{dist}(v_i, v_j). \end{aligned}$$

Determine a lower and an upper bound to the diameter of G as functions of R and Δ. Justify your answer.

Exercises (cont'd)

Exercise

Let $G = (V, E)$ be a connected, undirected graph with n nodes and let k be an integer, $1 \leq k < n$. For an arbitrary $v \in V$ consider its approximated closeness centrality $\tilde{c}(v)$ computed with the Eppstein and Wang's method, using k random pivots $v_1, v_2, \ldots, v_k$ drawn from V independently, with replacement and with uniform probability. Show that $1/\tilde{c}(v)$ is an unbiased estimator of $1/c(v)$, i.e.,

$$\mathrm{E}\left[\frac{1}{\tilde{c}(v)}\right] = \frac{1}{c(v)},$$

where $c(v)$ is the exact closeness centrality of v. For simplicity assume that $\tilde{c}(v)$ is computed as $k(n-1)/(n\sum_{i=1}^{k} \mathrm{dist}(v_i, v))$ even if v is a pivot.

Theory questions

- Consider a weighted connected graph $G = (V, E)$ and let C be a cycle in G. Argue that an edge $e \in C$ with weight strictly greater than any other edge of C cannot belong to any MST of G.
- Let $D(G)$ be the *diameter* of an undirected graph G. Define what $D(G)$ is and explain how an estimate $\Delta \in [D(G), 2D(G)]$ can be computed in linear time.
- What are the characteristics of nodes with high closeness centrality in a graph G?

References

L+11 S. Lattanzi et al. Filtering: a method for solving graph problems in MapReduce. ACM-SPAA 2011: 85-94

B+16 S. Bhagat et al. Three and a half degrees of separation. https://research.fb.com/three-and-a-half-degrees-of-separation/

BRV11 P. Boldi, M. Rosa, S. Vigna. HyperANF: approximating the neighbourhood function of very large graphs on a budget. WWW 2011: 625-634.

BV14 P. Boldi, S. Vigna. Axioms for Centrality. Internet Mathematics 10(3-4): 222-262 (2014)

EW04 D. Eppstein, J. Wang: Fast Approximation of Centrality. J. Graph Algorithms Appl. 8: 39-45 (2004)