MU5IN075
Network Analysis and Mining
2. Graph algorithms

Esteban Bautista-Ruiz, Lionel Tabourier

LIP6 – CNRS and Sorbonne Université

first_name.last_name@lip6.fr

September 21, 2021

Outline

Outline

Definitions and notations

A graph $G = (V, E)$ is a couple of sets.

- V is the set of *nodes*
- $E \subseteq (V \times V)$ is the set of *edges*

We denote :

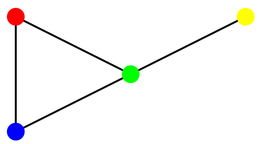
- $n = |V|$ the number of nodes
- $m = |E|$ the number of edges

u and v are neighbors if there is an edge between them.

Degree: (k_i or $d(i)$) number of neighbors of i

Average degree, density

- Average degree of a graph, $\overline{d} = \frac{\sum_v d(v)}{n}$
- Density of a graph, $\delta = \frac{2m}{n(n-1)}$



degrees : **2, 2, 3, 1** ; average degree 2

$n = 4$, $m = 4$, $\delta = \frac{8}{12} = 0.66..$

Connectedness

Path from u to v: sequence of edges
$(u, v_1), (v_1, v_2), \ldots, (v_{\alpha-1}, v)$
Length: number of edges in the path (here α)

Connected component: maximal set of nodes such that $\exists$ a path between any pair of nodes

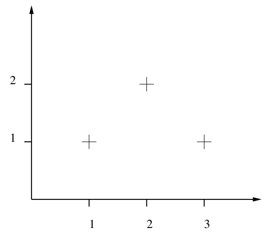
Connected graph: only one connected component

Distributions

Distribution : synthetic way to represent a sequence of values.
$\rightarrow$ how many times a value occurs in the sequence?

Example/reminder of the degree distribution:
4 nodes, degrees : 2 2 3 1

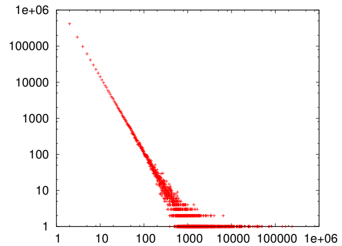
$1 \rightarrow 1$, $2 \rightarrow 2$, $3 \rightarrow 1$



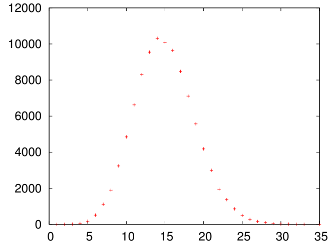
Degree distribution characteristics

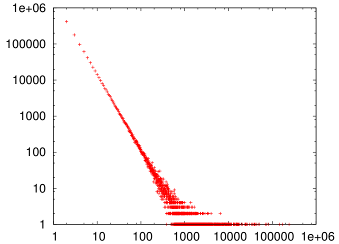
Benefit: characterize qualitatively a sequence of values.

Heterogeneous = non-homogeneous distribution (various types of behaviors), in practice often close to a power law



Heterogeneous vs homogeneous distributions





Homogeneous (e.g. distance distribution in a graph)

Idea of normality (and of exceptions)

Heterogeneous (e.g. degree distribution in a graph)

Any kind of behaviours → no notion of normality/exceptions

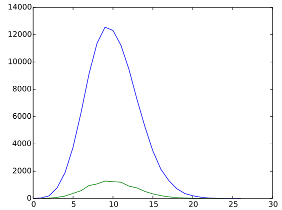
Two choices:

- N_k: number of occurrences of value k in the sequence
- p_k: proportion of the value k in the sequence
  → Normalized distribution

$$p_k = \frac{N_k}{n}$$

Just a change of the value on the Y-axis.

Allow to compare graphs with different sizes:



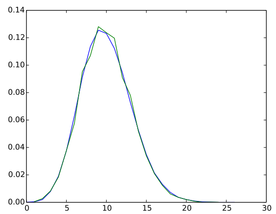
Two choices:

- N_k: number of occurrences of value k in the sequence
- p_k: proportion of the value k in the sequence
  → Normalized distribution

$$p_k = \frac{N_k}{n}$$

Just a change of the value on the Y-axis.

Allow to compare graphs with different sizes:

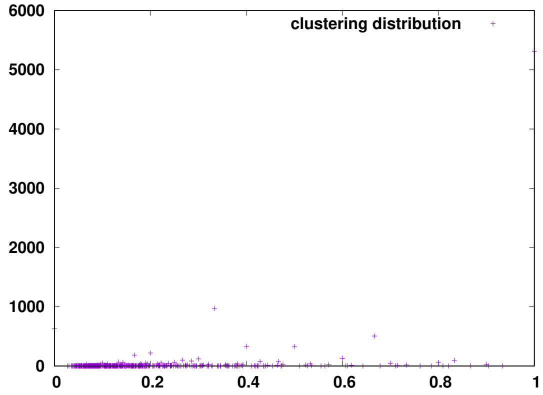


Notion of cumulative distributions

- Distribution of k:
  N_k: number of occurrences equal to k
- Cumulative distribution of k:
  C_k: number of occurrences lower or equal to k
- Inverse cumulative distribution of k:
  IC_k: number of occurrences greater or equal to k

Relevance of cumulative distributions

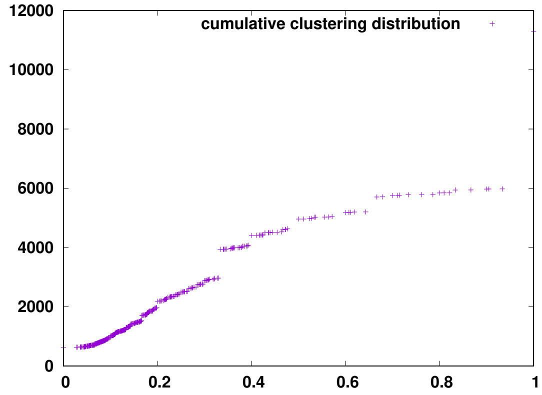
Illustration: clustering distribution of a coauthoring network



Definitions and metrics
Algorithms
Approximate measurements
Reminder
Distributions

Relevance of cumulative distributions

Illustration: clustering distribution of a coauthoring network

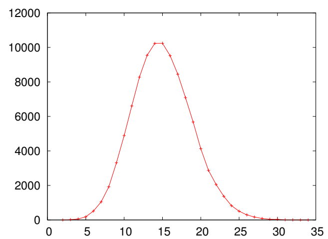


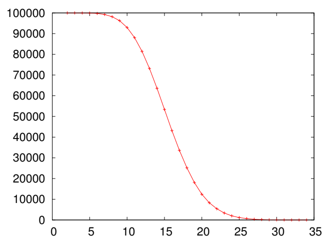
Definitions and metrics
Algorithms
Approximate measurements
Reminder
Distributions

Cumulative and inverse cumulative distribution

- N_k: number of occurrences equal to k
- C_k: number of occurrences lower or equal to k
- IC_k: number of occurrences greater or equal to k

N_k and IC_k for a homogeneous distribution:





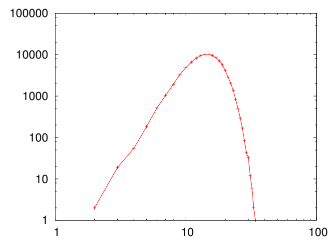
linear scale

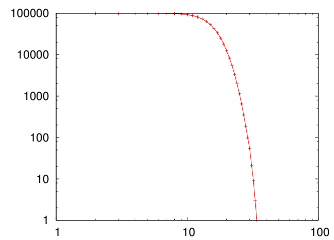
Definitions and metrics
Algorithms
Approximate measurements
Reminder
Distributions

Cumulative and inverse cumulative distribution

- N_k: number of occurrences equal to k
- C_k: number of occurrences lower or equal to k
- IC_k: number of occurrences greater or equal to k

N_k and IC_k for a homogeneous distribution:



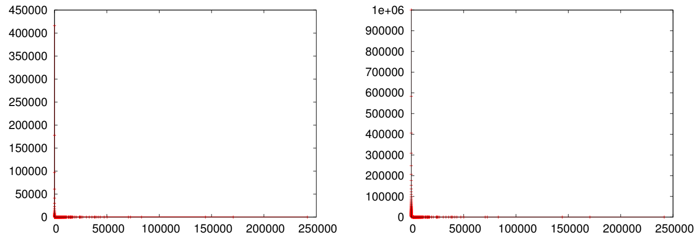


log-log scale

Cumulative and inverse cumulative distribution

- N_k: number of occurrences equal to k
- C_k: number of occurrences lower or equal to k
- IC_k: number of occurrences greater or equal to k

N_k and IC_k for a heterogeneous distribution:



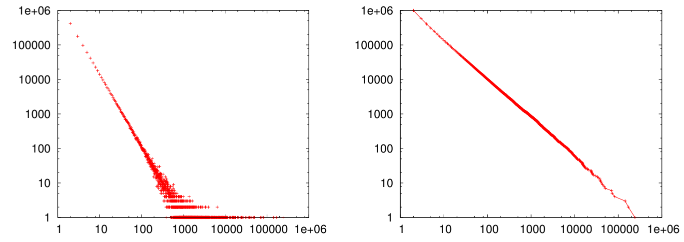
linear scale

Definitions and metrics
Algorithms
Approximate measurements
Reminder
Distributions

Cumulative and inverse cumulative distribution

- N_k: number of occurrences equal to k
- C_k: number of occurrences lower or equal to k
- IC_k: number of occurrences greater or equal to k

N_k and IC_k for a heterogeneous distribution:



log-log scale

Definitions and metrics
Algorithms
Approximate measurements
Reminder
Distributions

Cumulative and inverse cumulative distribution

- N_k: number of occurrences equal to k
- C_k: number of occurrences lower or equal to k
- IC_k: number of occurrences greater or equal to k

Homogeneous and heterogeneous

- can be distinguished on both normal and cumulative distributions

Ex: power-law

- $N_k \sim k^{-\alpha} \Longrightarrow C_k \sim k^{-\alpha+1}$
  Remark: same idea as $\int x^{-\alpha}\, dx \sim x^{-\alpha+1}$

Definitions and metrics
Algorithms
Approximate measurements
Reminder
Distributions

Cumulative and inverse cumulative distribution

- N_k: number of occurrences equal to k
- C_k: number of occurrences lower or equal to k
- IC_k: number of occurrences greater or equal to k

Homogeneous and heterogeneous

- can be distinguished on both normal and cumulative distributions

Ex: power-law

- $N_k \sim k^{-\alpha} \Longrightarrow C_k \sim k^{-\alpha+1}$
  Remark: same idea as $\int x^{-\alpha}\, dx \sim x^{-\alpha+1}$

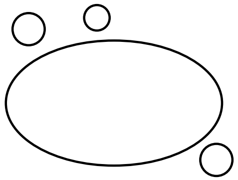
Outline

Definitions and metrics | Connected components
Algorithms | Distances computation
Approximate measurements | Local density

Connectedness (reminder)

For complex networks

In general, giant component
$\rightarrow$ contains most nodes



Q : How to identify the giant component? How to count the connected components?

Definitions and metrics | Connected components
Algorithms | Distances computation
Approximate measurements | Local density

Breadth First Search algorithm (BFS)

Parcours en largeur

Algorithm 1: Breadth First Search of a graph *G* from node *s*.

```
begin
    F ← CreateEmptyQueue()
    Enqueue(F,s)
    Mark(s)
    while F not empty do
        u ← DequeueFirstElement(F)
        Display u
        for v neighbor of u in G do
            if Unmarked(v) then
                Enqueue(F,v)
                Mark(v)
            end
        end
    end
end
```


Definitions and metrics | Connected components
Algorithms | Distances computation
Approximate measurements | Local density

Breadth First Search algorithm (BFS)

Properties of a BFS:

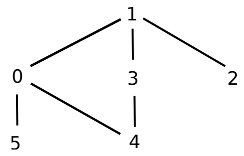
- From a node : we detect its connected component
  $\rightarrow$ 1 BFS per component
- Complexity: $\mathcal{O}(m)$
- With parentage memorization:
  spanning tree *(fr: arbre couvrant)* of shortest paths

Breadth First Search algorithm (BFS)

Properties of a BFS:

- From a node : we detect its connected component
  $\rightarrow$ 1 BFS per component
- Complexity: $\mathcal{O}(m)$
- With parentage memorization:
  spanning tree *(fr: arbre couvrant)* of shortest paths

Example



1. Apply the algorithm on the graph above, starting from node 3. Draw the corresponding BFS tree.
2. How to modify it so that it returns a tree of shortest paths from node s? Indicate the distances on the BFS tree.

Modified BFS

Algorithm 2: Distance from node s in graph G.

```
begin
    F ← CreateEmptyQueue()
    Enqueue(F,s)
    ∀v Dist(v) initialized at -1
    Dist(s) ← 0
    while F not empty do
        u ← DequeueFirstElement(F)
        Display u
        for v neighbor of u in G do
            if Dist(v) = -1 then
                Enqueue(F,v)
                Dist(v) ← Dist(u) +1
            end
        end
    end
end
```


Distances computation

Distance from a node to all others:
modified breadth first search $\rightarrow$ Complexity: $\mathcal{O}(m)$

Average distance, diameter

Need all distances $\rightarrow \mathcal{O}(nm)$
possible to approximate or to give bounds on the diameter
More info about this in *Approximate measurements*

Distances computation

Distance from a node to all others:
modified breadth first search → Complexity: $\mathcal{O}(m)$

Average distance, diameter

Need all distances $\to \mathcal{O}(nm)$
possible to approximate or to give bounds on the diameter
More info about this in *Approximate measurements*

Going back to local density

Several means to capture this idea, for example:

- clustering coefficient: $\mathtt{cc}(G) = \frac{\sum_v \frac{\Delta(v)}{\wedge(v)}}{n'}$
  n' = # nodes with degree ≥ 2
- transitive ratio: $\mathtt{tr}(G) = \frac{3\Delta(G)}{\wedge(G)}$

In other words:

- clustering coefficient: compute a value for each node (with degree ≥ 2), then average
- transitive ratio: direct computation

Going back to local density

Several means to capture this idea, for example:

- clustering coefficient: $\mathtt{cc}(G) = \frac{\sum_v \frac{\Delta(v)}{\wedge(v)}}{n'}$
  n' = # nodes with degree ≥ 2
- transitive ratio: $\mathtt{tr}(G) = \frac{3\Delta(G)}{\wedge(G)}$

In other words:

- clustering coefficient: compute a value for each node (with degree ≥ 2), then average
- transitive ratio: direct computation

Transitive ratio *vs* Clustering coefficient

We have $n \in \mathbb{N}$, let's G_n be the graphs of $2n + 1$ nodes and $3n$ edges such that:

- a unique node n_0 is connected to all other nodes in the network
- all other nodes have degree 2

Exercise :

1. Draw the cases of G_3, G_4.
2. Compute the local density coefficients for G_4.
3. How do these coefficients evolve when n goes to ∞?
4. Deduce how to interpret these coefficients in terms of probability.

Computing the number of triangles

Both coefficients rely on the number of triangles. How to enumerate the number of triangles that node n belongs to?

Naive answer: for all nodes v, for any pair of neighbors (u_1, u_2) of v, test if (u_1, u_2) exists.

Algorithm 3: Naive triangle counting algorithm

```
for v ∈ V do
    for u1 ∈ N(v) do
        for u2 ∈ N(v), u2 ≠ u1 do
            if u1 ∈ N(u2) then
                nb++
            end
        end
    end
end
return nb/6
```

note: $N(x)$ is the list of neighbors of node x

Definitions and metrics | Connected components
Algorithms | Distances computation
Approximate measurements | Local density

Computing the number of triangles

Both coefficients rely on the number of triangles. How to enumerate the number of triangles that node n belongs to?

Naive answer: for all nodes v, for any pair of neighbors (u_1, u_2) of v, test if (u_1, u_2) exists.

Questions:

1. Why dividing by 6? Because 1 triangle is seen 6 times
2. Time complexity of the algorithm? $\Sigma_v \frac{d(v).(d(v)-1)}{2}$
3. In which case is it expensive? if $d(v)$ is large
4. Is it a problem for the networks we are working on? yes, because heterogeneous degree distribution

Definitions and metrics | Connected components
Algorithms | Distances computation
Approximate measurements | Local density

Computing the number of triangles

Both coefficients rely on the number of triangles. How to enumerate the number of triangles that node n belongs to?

Naive answer: for all nodes v, for any pair of neighbors (u_1, u_2) of v, test if (u_1, u_2) exists.

Questions:

1. Why dividing by 6? Because 1 triangle is seen 6 times
2. Time complexity of the algorithm? $\Sigma_v \frac{d(v).(d(v)-1)}{2}$
3. In which case is it expensive? if $d(v)$ is large
4. Is it a problem for the networks we are working on? yes, because heterogeneous degree distribution

Definitions and metrics | Connected components
Algorithms | Distances computation
Approximate measurements | Local density

Computing the number of triangles

Both coefficients rely on the number of triangles. How to enumerate the number of triangles that node n belongs to?

Naive answer: for all nodes v, for any pair of neighbors (u_1, u_2) of v, test if (u_1, u_2) exists.

Questions:

1. Why dividing by 6? Because 1 triangle is seen 6 times
2. Time complexity of the algorithm? $\Sigma_v \frac{d(v).(d(v)-1)}{2}$
3. In which case is it expensive? if $d(v)$ is large
4. Is it a problem for the networks we are working on? yes, because heterogeneous degree distribution

Computing the number of triangles

Both coefficients rely on the number of triangles. How to enumerate the number of triangles that node n belongs to?

Naive answer: for all nodes v, for any pair of neighbors (u_1, u_2) of v, test if (u_1, u_2) exists.

Questions:

1. Why dividing by 6? Because 1 triangle is seen 6 times
2. Time complexity of the algorithm? $\Sigma_v \frac{d(v).(d(v)-1)}{2}$
3. In which case is it expensive? if $d(v)$ is large
4. Is it a problem for the networks we are working on? yes, because heterogeneous degree distribution

Definitions and metrics | Connected components
Algorithms | Distances computation
Approximate measurements | Local density

Computing the number of triangles

Both coefficients rely on the number of triangles. How to enumerate the number of triangles that node n belongs to?

Naive answer: for all nodes v, for any pair of neighbors (u_1, u_2) of v, test if (u_1, u_2) exists.

Questions:

1. Why dividing by 6? Because 1 triangle is seen 6 times
2. Time complexity of the algorithm? $\Sigma_v \frac{d(v).(d(v)-1)}{2}$
3. In which case is it expensive? if $d(v)$ is large
4. Is it a problem for the networks we are working on? yes, because heterogeneous degree distribution

23/32

Improved computation of the number of triangles

Other point view:
Consider edge (u, v), how many triangles does it belong to?
$\rightarrow |N(u) \cap N(v)|$

Definitions and metrics | Connected components
Algorithms | Distances computation
Approximate measurements | Local density

Improved computation of the number of triangles

Other point view:
Consider edge (u, v), how many triangles does it belong to?
$\rightarrow |N(u) \cap N(v)|$

24/32

Improved computation of the number of triangles

Other point view:
Consider edge (u, v), how many triangles does it belong to?
$\rightarrow |N(u) \cap N(v)|$

Algorithm 4: Improved triangle counting algorithm

for $(u, v) \in E$, $u < v$ **do**
  for $w \in N(u) \cap N(v)$ **do**
    if $v < w$ **then**
      nb++
    end
  end
end
return nb

notes:

- we added the inequalities to count each triangle once
- time complexity in $\sum_{(u,v)\in E} d(u).d(v)$ and can be improved, how?
- practical running time better than naive version

Improved computation of the number of triangles

Other point view:
Consider edge (u, v), how many triangles does it belong to?
$\rightarrow |N(u) \cap N(v)|$

Algorithm 5: Improved triangle counting algorithm

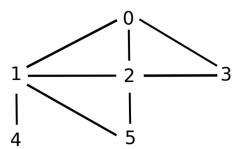
for $(u, v) \in E$, $u < v$ **do**
  for $w \in N(u) \cap N(v)$ **do**
    if $v < w$ **then**
      nb++
    end
  end
end
return nb

notes:

- we added the inequalities to count each triangle once
- time complexity in $\sum_{(u,v)\in E} d(u).d(v)$ and can be improved, how?
- practical running time better than naive version

Exercise

Apply the previous algorithm to the following graph:



Higher order cliques

Clique

Triangles are 3-nodes cliques. A clique is a complete subgraph.

- subgraph: graph obtained considering a subset of nodes and the edges between these nodes *(fr: sous-graphe)*
- complete: any node is connected to all others *(fr: complet)*

Maximal cliques

Decomposition of a graph into its maximal cliques

Obtaining the list of all maximal cliques is known to be computationally hard
$\rightarrow$ we favor search with fixed size

Higher order cliques

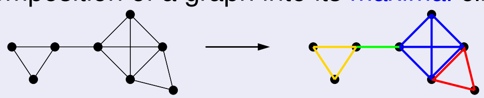
Clique

Triangles are 3-nodes cliques. A clique is a complete subgraph.

- subgraph: graph obtained considering a subset of nodes and the edges between these nodes *(fr: sous-graphe)*
- complete: any node is connected to all others *(fr: complet)*

Maximal cliques

Decomposition of a graph into its maximal cliques



Obtaining the list of all maximal cliques is known to be computationally hard
$\rightarrow$ we favor search with fixed size

Outline

Approximations

Approximation: given a property P, how to estimate this property on a given graph.
Examples: average degree, average distance, diameter, ...

One possible approach (sampling)

1. Pick a node v of G at random
2. Estimate the property for v
3. Go back to step 1 while the estimation is not good enough

Questions:

- How to express the notion of "good enough"?
- How to know if this approach provides a good approximation or not?

Approximations

Approximation: given a property P, how to estimate this property on a given graph.
Examples: average degree, average distance, diameter, ...

One possible approach (sampling)

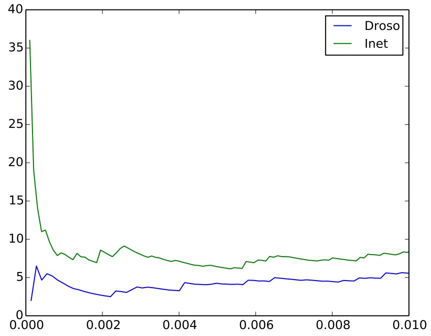
1. Pick a node v of G at random
2. Estimate the property for v
3. Go back to step 1 while the estimation is not good enough

Questions:

- How to express the notion of "good enough"?
- How to know if this approach provides a good approximation or not?

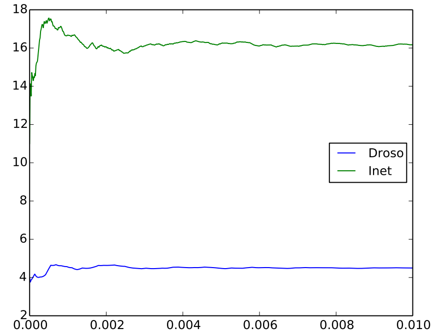
Average degree

Application of the former method to the average degree:
X-axis : fraction of measured values



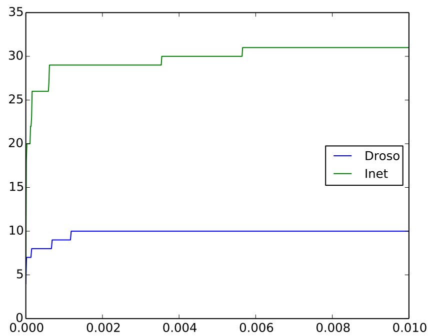
Average distance

Application of the former method to the average degree:
X-axis : fraction of measured values



Diameter

Application of the former method to the average degree:
X-axis : fraction of measured values



Approximations

Quality of the approximation: depends on the nature of the property.

Other possible approach

- Compute (lower and upper) bounds of the property
- Rely on the property to drive the computations

Example: For every node v, let max_v be the greatest distance from v to a node of G. Then the diameter D of G is such that:

$$max_v \leq D \leq 2max_v$$

Exercise : explain why.

Approximations

Quality of the approximation: depends on the nature of the property.

Other possible approach

- Compute (lower and upper) bounds of the property
- Rely on the property to drive the computations

Example: For every node v, let max_v be the greatest distance from v to a node of G. Then the diameter D of G is such that:

$$max_v \leq D \leq 2max_v$$

Exercise : explain why.