Graph Mining Algorithms

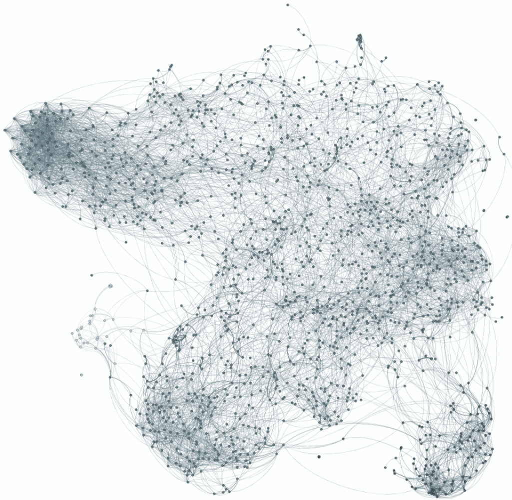
Mauro Sozio

Telecom ParisTech

March 20, 2015

Finding dense regions in a graph..

for community detection, spam detection, event detection...



Graph: Definitions

Definition ((Undirected) Graph)

A graph G is a pair (V_G, E_G), where V_G is a set of *nodes*, while E_G is a set of *edges* (u, v) with $u, v \in V_G$.

Other important definitions:

Graph: Definitions

Definition ((Undirected) Graph)

A graph G is a pair (V_G, E_G), where V_G is a set of *nodes*, while E_G is a set of *edges* (u, v) with $u, v \in V_G$.

Other important definitions:

- A graph $H = (V_H, E_H)$ is a (induced) subgraph of $G = (V_G, E_G)$ if the following two conditions hold: $V_H \subseteq V_G$, moreover, $(u, v) \in E_H$ if and only if $u, v \in H$ and $(u, v) \in E_G$.

Graph: Definitions

Definition ((Undirected) Graph)

A graph G is a pair (V_G, E_G), where V_G is a set of *nodes*, while E_G is a set of *edges* (u, v) with $u, v \in V_G$.

Other important definitions:

- A graph $H = (V_H, E_H)$ is a (induced) subgraph of $G = (V_G, E_G)$ if the following two conditions hold: $V_H \subseteq V_G$, moreover, $(u, v) \in E_H$ if and only if $u, v \in H$ and $(u, v) \in E_G$.
- $\delta_G(v)$ denotes the number of edges incident to v in G, while $\delta_H(v)$ denotes the number of edges incident to v in H.

Density of a graph

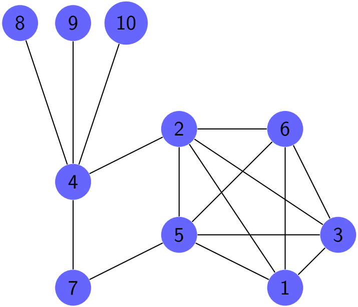
Definition (average degree density)

Given a graph $G = (E_G, V_G)$ its (average degree) density $\rho(G)$ is defined as $\rho(G) = \frac{|E_G|}{|V_G|}$.

Definition (clique density)

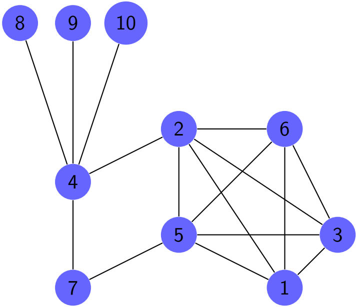
Given a graph $G = (E_G, V_G)$ its (clique) density $\phi(G)$ is defined as $\phi(G) = \frac{2 \cdot |E_G|}{|V_G| \cdot (|V_G| - 1)}$.

Example



$H = (\{4, 8, 9, 10\}, \{(4, 8)(4, 9)(4, 10)\})$
$\delta_G(4) = 5, \delta_H(4) = 3$

Example



$H = (\{4, 8, 9, 10\}, \{(4, 8)(4, 9)(4, 10)\})$
$\delta_G(4) = 5, \delta_H(4) = 3$
$\rho(G) = \frac{16}{10}, \rho(H) = \frac{3}{4}, \phi(H) = \frac{2 \cdot 3}{12}$

Simple lemma

Lemma

Given a graph $G = (V_G, E_G)$, we have:

$$\sum_{v \in V_G} \delta_G(v) = 2|E(G)|.$$

Proof.

Every edge $(u, v) \in E(G)$ is counted exactly twice in the summation: Once in $\delta_G(u)$ and the second time in $\delta_G(v)$. $\square$

Our main problem

Definition (Densest subgraph problem)

Given a graph $G = (V_G, E_G)$, find a subgraph H of G with maximum average degree density.

Facts: A global optimum can be computed in polynomial time. There is a linear-time algorithm that computes an approximation to the problem..

Densest Subgraph Algorithm

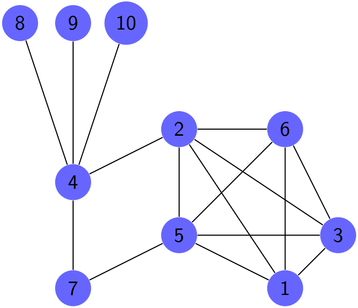
$H = G$;

while (G contains at least one edge)

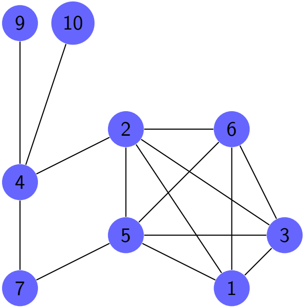
- let v be the node with minimum degree $\delta_G(v)$ in G;
- remove v and all its edges from G;
- if $\rho(G) > \rho(H)$ then $H \leftarrow G$;

return H;

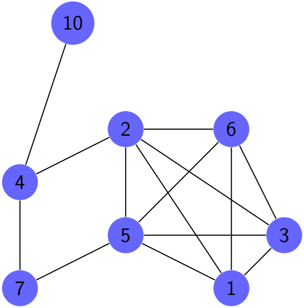
Densest Subgraph Algorithm:Example



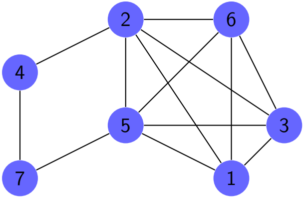
Densest Subgraph Algorithm:Example



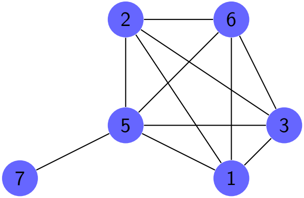
Densest Subgraph Algorithm:Example



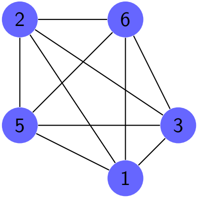
Densest Subgraph Algorithm:Example



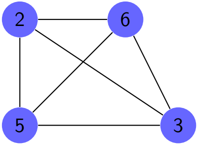
Densest Subgraph Algorithm:Example



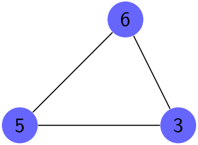
Densest Subgraph Algorithm:Example



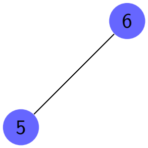
Densest Subgraph Algorithm:Example



Densest Subgraph Algorithm:Example

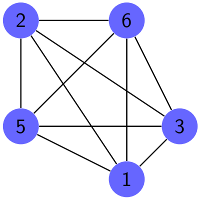


Densest Subgraph Algorithm:Example



Densest Subgraph Algorithm:Example

Densest Subgraph Algorithm:Example



Approximation Guarantee

Theorem

Let O be a densest subgraph in G. Our algorithm finds a subgraph H s.t.

$$\rho(H) \geq \frac{\rho(O)}{2}.$$

Approximation Guarantee

Lemma

Let O be a densest subgraph in G, then:

$$\forall v \in V_O \quad \delta_G(v) \geq \rho(O).$$

Proof.

We show that if there is v in G with $\delta_G(v) < \rho(O)$, then O is not densest.

$$\rho(O \setminus \{v\}) = \frac{|E_O| - \delta_G(v)}{|V_O| - 1}$$

□

Approximation Guarantee

Lemma

Let O be a densest subgraph in G, then:

$$\forall v \in V_O \quad \delta_G(v) \geq \rho(O).$$

Proof.

We show that if there is v in G with $\delta_G(v) < \rho(O)$, then O is not densest.

$$\begin{aligned}\rho(O \setminus \{v\}) &= \frac{|E_O| - \delta_G(v)}{|V_O| - 1} \\ &> \frac{|E_O| - \rho(O)}{|V_O| - 1}\end{aligned}$$

□

Approximation Guarantee

Lemma

Let O be a densest subgraph in G, then:

$$\forall v \in V_O \quad \delta_G(v) \geq \rho(O).$$

Proof.

We show that if there is v in G with $\delta_G(v) < \rho(O)$, then O is not densest.

$$\begin{aligned}
\rho(O \setminus \{v\}) &= \frac{|E_O| - \delta_G(v)}{|V_O| - 1} \\
&> \frac{|E_O| - \rho(O)}{|V_O| - 1} \\
&= \frac{|V_O|\rho(O) - \rho(O)}{|V_O| - 1} = \rho(O)\frac{|V_O| - 1}{|V_O| - 1} = \rho(O).
\end{aligned}$$

□

Approximation Guarantee

Theorem

Let G be **any** *undirected graph. Let O be* **a** *densest subgraph of G, while let H be the subgraph computed by our algorithm with input G. Then,*

$$\rho(H) \geq \frac{\rho(O)}{2}.$$

Proof of the approx. guarantee

Proof.

Consider the first step in the algo such that $v \in V_O$ has minimum degree in the current graph $G = (V_G, E_G)$. From the previous lemma $\delta_G(v) \geq \delta_O(v) \geq \rho(O)$. Hence,

$$\rho(H) \geq \rho(G)$$

Proof of the approx. guarantee

Proof.

Consider the first step in the algo such that $v \in V_O$ has minimum degree in the current graph $G = (V_G, E_G)$. From the previous lemma $\delta_G(v) \geq \delta_O(v) \geq \rho(O)$. Hence,

$$\begin{aligned}\rho(H) &\geq \rho(G) \\ &= \frac{|E_G|}{|V_G|}\end{aligned}$$

Proof of the approx. guarantee

Proof.

Consider the first step in the algo such that $v \in V_O$ has minimum degree in the current graph $G = (V_G, E_G)$. From the previous lemma $\delta_G(v) \geq \delta_O(v) \geq \rho(O)$. Hence,

$$\begin{aligned}\rho(H) &\geq \rho(G) \\ &= \frac{|E_G|}{|V_G|} \\ &= \frac{\frac{1}{2} \cdot \sum_{v \in V_G} \delta_G(v)}{|V_G|}\end{aligned}$$

Proof of the approx. guarantee

Proof.

Consider the first step in the algo such that $v \in V_O$ has minimum degree in the current graph $G = (V_G, E_G)$. From the previous lemma $\delta_G(v) \geq \delta_O(v) \geq \rho(O)$. Hence,

$$\begin{aligned}
\rho(H) &\geq \rho(G) \\
&= \frac{|E_G|}{|V_G|} \\
&= \frac{\frac{1}{2} \cdot \sum_{v \in V_G} \delta_G(v)}{|V_G|} \\
&\geq \frac{1}{2} \cdot \frac{|V_G| \rho(O)}{|V_G|}
\end{aligned}$$

Proof of the approx. guarantee

Proof.

Consider the first step in the algo such that $v \in V_O$ has minimum degree in the current graph $G = (V_G, E_G)$. From the previous lemma $\delta_G(v) \geq \delta_O(v) \geq \rho(O)$. Hence,

$$\begin{aligned}
\rho(H) &\geq \rho(G) \\
&= \frac{|E_G|}{|V_G|} \\
&= \frac{\frac{1}{2} \cdot \sum_{v \in V_G} \delta_G(v)}{|V_G|} \\
&\geq \frac{1}{2} \cdot \frac{|V_G|\rho(O)}{|V_G|} \\
&= \frac{\rho(O)}{2}.
\end{aligned}$$

Running time

Our algo can be implemented in linear time in the size of the input graph (i.e. the total number of edges and nodes in the graph).

- for each value δ in $[1, n]$ maintain a list of nodes with degree δ in the current graph.
- As nodes are removed from the graph, update the lists so that each node is placed in the correct list (depending on its current degree).

A Parallel Algorithm for Densest Subgraph

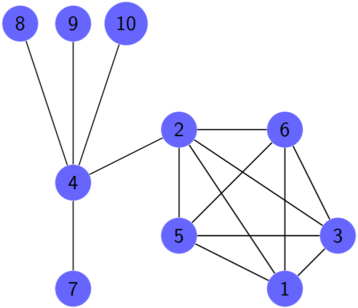
Require: an undirected graph G, a value $\epsilon > 0$

$H = G$;
while (G contains at least one edge)

- rem. all nodes v (and their edges) with $\delta_G(v) \leq 2(1+\epsilon)\rho(G)$ from G.
- if $\rho(G) > \rho(H)$ then $H \leftarrow G$;

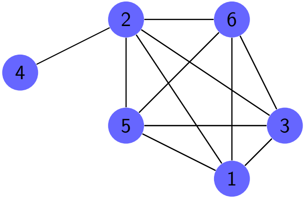
return H;

Parallel algorithm: Example



$\epsilon = 0.1$
Iteration 1:
$\rho(G) = \frac{16}{10}$, remove nodes with degree $\leq 2 * (1.1) * 1.6 = 3.52$.

Parallel algorithm: Example

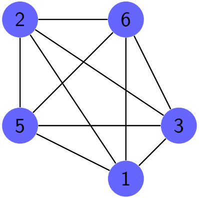


$\epsilon = 0.1$

Iteration 2:

$\rho(G) = \frac{11}{6}$, remove nodes with degree $\leq 2 * (1.1) * \frac{11}{6} = 3.45$.

Parallel algorithm: Example

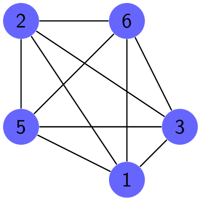


$\epsilon = 0.1$
Iteration 3:
$\rho(G) = \frac{10}{5}$, remove nodes with degree $\leq 2 * (1.1) * 2 = 4.4$.

Parallel algorithm: Example

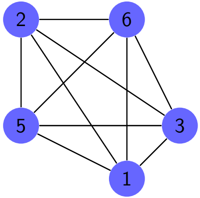
$\epsilon = 0.1$
Iteration 4:
Empty Graph.

Parallel algorithm: Example



$\epsilon = 0.1$
$2(1+\epsilon)-$**Approx. Densest Subgraph!**

Parallel algorithm: Example



$\epsilon = 0.1$
$2(1+\epsilon)-$**Approx. Densest Subgraph!**
What if ϵ is large? (say $\epsilon = 0.5$)

Approx. guarantee of the parallel algo

Theorem

Let $O = (V_O, E_O)$ be a densest subgraph and let $H = (V_H, E_H)$ be the subgraph found by our algo, with parameter $\epsilon > 0$. Then, $\rho(H) \geq \frac{\rho(O)}{2(1+\epsilon)}$.

Proof.

Let $O = (V_O, E_O)$ be a densest subgraph. Consider the first step in the algo such that we remove a node $v \in V_O$ from the current graph G (there must be such a step). From Lemma 7, $\delta_G(v) \geq \delta_O(v) \geq \rho(O)$. Hence,

$$\rho(O) \leq \delta_G(v)$$

Approx. guarantee of the parallel algo

Theorem

Let $O = (V_O, E_O)$ be a densest subgraph and let $H = (V_H, E_H)$ be the subgraph found by our algo, with parameter $\epsilon > 0$. Then, $\rho(H) \geq \frac{\rho(O)}{2(1+\epsilon)}$.

Proof.

Let $O = (V_O, E_O)$ be a densest subgraph. Consider the first step in the algo such that we remove a node $v \in V_O$ from the current graph G (there must be such a step). From Lemma 7, $\delta_G(v) \geq \delta_O(v) \geq \rho(O)$. Hence,

$$\begin{aligned} \rho(O) &\leq \delta_G(v) \\ &\leq 2(1+\epsilon)\rho(G) \\ &\leq 2(1+\epsilon)\rho(H) \end{aligned}$$

□

Running time of the parallel algo

Theorem

The number of iterations of the parallel algo with input $G = (V_G, E_G)$ *and* $\epsilon > 0$ *is at most* $\lceil \log_{1+\epsilon}(|V_G|) \rceil$.

Proof.

Consider any step t of the algo and let $G_t = (V_{G_t}, E_{G_t})$ be the subgraph at the beginning of that step. Let R_t be the set of nodes removed at the end of such step, i.e. the degree of any node in R_t is $\leq 2(1+\epsilon)\rho(G_t)$. Then,

$$2|E_{G_t}| = \sum_{v \in R_t} \delta_{G_t}(v) + \sum_{v \in V_{G_t} \setminus R_t} \delta_{G_t}(v)$$

Running time of the parallel algo

Theorem

The number of iterations of the parallel algo with input $G = (V_G, E_G)$ *and* $\epsilon > 0$ *is at most* $\lceil \log_{1+\epsilon}(|V_G|) \rceil$.

Proof.

Consider any step t of the algo and let $G_t = (V_{G_t}, E_{G_t})$ be the subgraph at the beginning of that step. Let R_t be the set of nodes removed at the end of such step, i.e. the degree of any node in R_t is $\leq 2(1+\epsilon)\rho(G_t)$. Then,

$$\begin{aligned} 2|E_{G_t}| &= \sum_{v \in R_t} \delta_{G_t}(v) + \sum_{v \in V_{G_t} \setminus R_t} \delta_{G_t}(v) \\ &> 2(1+\epsilon)(|V_{G_t}| - |R_t|)\rho(G_t) \end{aligned}$$

Running time of the parallel algo

Theorem

The number of iterations of the parallel algo with input $G = (V_G, E_G)$ and $\epsilon > 0$ is at most $\lceil \log_{1+\epsilon}(|V_G|) \rceil$.

Proof.

Consider any step t of the algo and let $G_t = (V_{G_t}, E_{G_t})$ be the subgraph at the beginning of that step. Let R_t be the set of nodes removed at the end of such step, i.e. the degree of any node in R_t is $\leq 2(1+\epsilon)\rho(G_t)$. Then,

$$\begin{aligned} 2|E_{G_t}| &= \sum_{v \in R_t} \delta_{G_t}(v) + \sum_{v \in V_{G_t} \setminus R_t} \delta_{G_t}(v) \\ &> 2(1+\epsilon)(|V_{G_t}| - |R_t|)\rho(G_t) \\ &= 2(1+\epsilon)(|V_{G_t}| - |R_t|)\frac{|E_{G_t}|}{|V_{G_t}|}. \end{aligned}$$

Running time of the parallel algo...

Proof.

Then,

$\square$

Running time of the parallel algo...

Proof.

Then,

$$2|E_{G_t}| > 2(1+\epsilon)(|V_{G_t}| - |R_t|)\frac{|E_{G_t}|}{|V_{G_t}|}, \qquad \Leftrightarrow$$

$\square$

Running time of the parallel algo...

Proof.

Then,

$$
\begin{aligned}
2|E_{G_t}| > 2(1+\epsilon)(|V_{G_t}| - |R_t|)\frac{|E_{G_t}|}{|V_{G_t}|}, &\qquad \Leftrightarrow \\
|V_{G_t}| > (1+\epsilon)(|V_{G_t}| - |R_t|), &\qquad \Leftrightarrow
\end{aligned}
$$

□

Running time of the parallel algo...

Proof.

Then,

$$
\begin{aligned}
2|E_{G_t}| &> 2(1+\epsilon)(|V_{G_t}| - |R_t|)\frac{|E_{G_t}|}{|V_{G_t}|}, && \Leftrightarrow \\
|V_{G_t}| &> (1+\epsilon)(|V_{G_t}| - |R_t|), && \Leftrightarrow
\end{aligned}
$$

$$|V_{G_{t+1}}| = |V_{G_t}| - |R_t| < \frac{|V_{G_t}|}{1+\epsilon}.$$

□

Running time of the parallel algo...

Proof.

Then,

$$
\begin{aligned}
2|E_{G_t}| > 2(1+\epsilon)(|V_{G_t}| - |R_t|)\frac{|E_{G_t}|}{|V_{G_t}|}, \qquad &\Leftrightarrow \\
|V_{G_t}| > (1+\epsilon)(|V_{G_t}| - |R_t|), \qquad &\Leftrightarrow \\
|V_{G_{t+1}}| = |V_{G_t}| - |R_t| < \frac{|V_{G_t}|}{1+\epsilon}. &
\end{aligned}
$$

Therefore $|V_{G_t}| \leq 1$ in $\leq t$ steps for any t such that $\frac{|V_G|}{(1+\epsilon)^t} \leq 1$, in particular when $t = \lceil \log_{1+\epsilon} |V_G| \rceil$. $\square$

k-cores

Definition (k-core)

Given a graph G and $k \geq 0$, a subgraph H of G is a k-core, if

- for every node $v \in V_H$, $\delta_H(v) \geq k$;
- the number of nodes V_H is maximized.

k-cores

Definition (k-core)

Given a graph G and $k \geq 0$, a subgraph H of G is a k-core, if

- for every node $v \in V_H$, $\delta_H(v) \geq k$;
- the number of nodes V_H is maximized.

A k-core can be computed in linear time in $|E_G|$ as follows.

While (at least one node has degree $< k$)

- remove all nodes with degree $< k$ from the current graph.

k-core decomposition

Note: a k-core might not be connected and is unique.

k-core decomposition

Note: a k-core might not be connected and is unique.

Note: If v belongs to a $k-$core then it belongs to a $\bar{k}-$core, $\bar{k} < k$.

k-core decomposition

Note: a k-core might not be connected and is unique.

Note: If v belongs to a $k-$core then it belongs to a $\bar{k}-$core, $\bar{k} < k$.

Definition (k-core decomposition)

A $k-$core decomposition of a graph G specifies for each node v in G an integer k_v such that v is in the k_v-core and k_v is maximum.

k-core decomposition

Note: a k-core might not be connected and is unique.

Note: If v belongs to a $k-$core then it belongs to a $\bar{k}-$core, $\bar{k} < k$.

Definition (k-core decomposition)

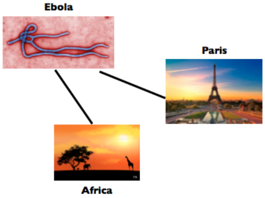
A $k-$core decomposition of a graph G specifies for each node v in G an integer k_v such that v is in the k_v-core and k_v is maximum.

Note: A $k-$core decomposition can be computed in linear time.

From Tweets to Dense Subgraphs

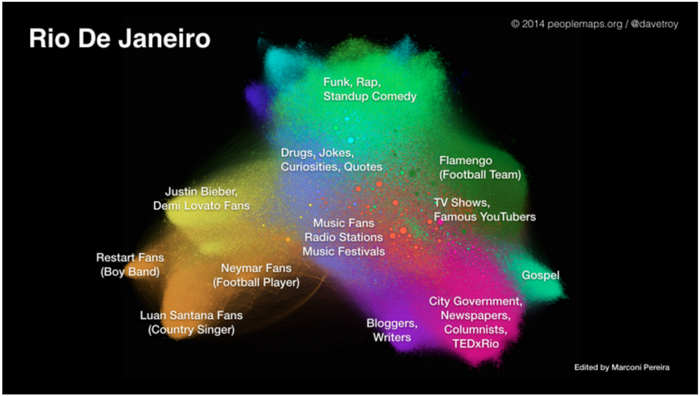
- Ebola à Paris
- Virus Ebola in France: false alarm
- De l'Afrique à Paris...

.
.
.

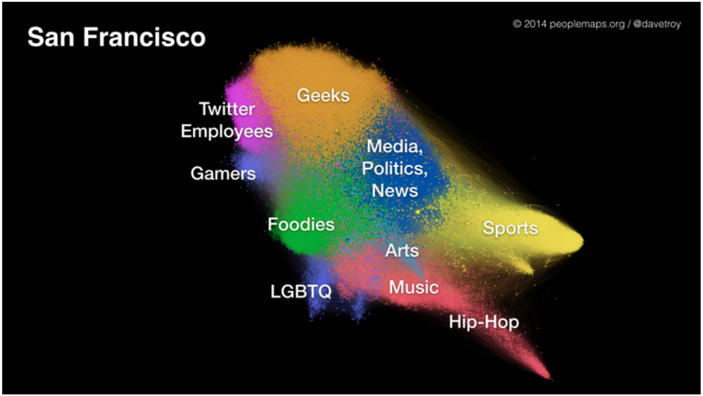


During the false alarm of the virus ebola in paris, `Paris`, `Ebola`, `Africa` co-occurred often in tweets. Can we find such events automatically?

Communities in Rio



Communities in San Francisco



Communities in Munich