Apparition de la composante géante pour un hypergraphe aléatoire

E. Coupechoux - M. Lelarge

INRIA-ENS

23 mars 2010

Outline

1. Giant component for random graphs
   - $G(n,p)$
   - $G(n,(d_i)_1^n)$
2. Hypergraphs and branching process approximation
   - Hypergraphs
   - Random hypergraphs
   - Branching process approximation
3. Giant component for random hypergraphs
   - Result
   - Exploring process
   - Differential equation approximation for Markov chains

Erdős-Rényi graphs

[Erdős and Rényi, *On the evolution of random graphs*, Publ. Math. Inst. Hungar. Acad. Sci., 1960]

- **Model :**
  $G(n,p)$ with $p = \dfrac{c}{n}$
  $C_1(n)$= largest connected component of $G\left(n, \dfrac{c}{n}\right)$
- **Sub-critical phase :** $c \leq 1$ (no giant component)

$$|C_1(n)| \underset{n\to\infty}{=} o_p(n)$$

- **Super-critical phase :** $c > 1$ (giant component of order n)

$$\exists \rho > 0, |C_1(n)| \underset{n\to\infty}{=} \rho n + o_p(n)$$

Graphs with given degree sequence $(d_i)_1^n$

[Molloy and Reed, *A critical point for random graphs with a given degree sequence*, Rand. Struct. Alg., 1995]

- **Model :**
  - For each $n \in \mathbb{N}$, $(d_i)_1^n$ sequence of non-negative integers such that there exists a graph with degree sequence $(d_i)_1^n$
  - $G(n, (d_i)_1^n)$ random graph with degree sequence $(d_i)_1^n$, uniformly chosen among all possibilities
- **Conditions :**
  $\exists (p_k)_{k=1}^{\infty}$ probability distribution such that :

  (i) $\sharp\{i : d_i = k\}/n \to p_k$ as $n \to \infty$, for every $k \geq 0$

  (ii) $\sum_k kp_k \in (0; \infty)$

  (iii) $p_1 > 0$

  (iv) $\sum_i d_i^2 = O(n)$

Graphs with given degree sequence $(d_i)_1^n$

$D \sim (p_k)_{k=1}^{\infty}$
$C_1(n)$= largest connected component of $G\left(n,(d_i)_1^n\right)$

- **Theorem :**
  - **Sub-critical phase :** $\mathbb{E}\left[D(D-1)\right] \leq \mathbb{E}\left[D\right]$
    (no giant component)

$$|C_1(n)| \underset{n\to\infty}{=} o_p(n)$$

  - **Super-critical phase :** $\mathbb{E}\left[D(D-1)\right] > \mathbb{E}\left[D\right]$
    (giant component of order n)

$$\exists \rho > 0, |C_1(n)| \underset{n\to\infty}{=} \rho n + o_p(n)$$

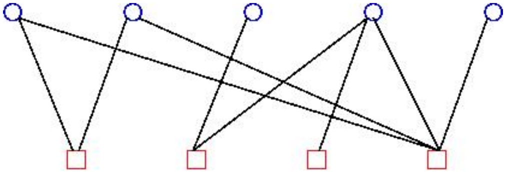
Hypergraph : definition

V and E finite sets

Hypergraph : $\gamma \subset V \times E$

$V = \{\text{vertices}\}$

$E = \{\text{hyper-edges}\}$



Degree of a vertex v = its number of edges

Weight of a hyper-edge e = its number of edges

- Degree and weight functions :

$$\mathbf{d} : \begin{cases} V & \longrightarrow & \mathbb{N} \\ v & \longmapsto & \mathbf{d}(v) = \text{degree of } v \end{cases}$$

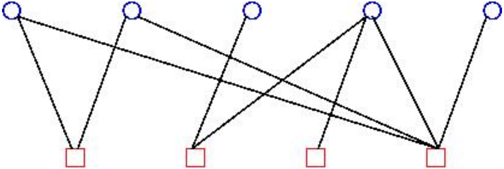
$$\mathbf{w} : \begin{cases} E & \longrightarrow & \mathbb{N} \\ e & \longmapsto & \mathbf{w}(e) = \text{weight of } e \end{cases}$$

- Degree and weight frequency vectors :

  $\mathbf{p} = (p_1, ..., p_L) : p_d =$ number of vertices of degree d
  $\mathbf{q} = (q_1, ..., q_L) : q_w =$ number of hyper-edges of weight w

- Correspondence

  $\mathbf{p} = (|\mathbf{d}^{-1}(\{1\})|, ..., |\mathbf{d}^{-1}(\{L\})|) = \mathbf{n}(\mathbf{d})$
  $\mathbf{q} = (|\mathbf{w}^{-1}(\{1\})|, ..., |\mathbf{w}^{-1}(\{L\})|) = \mathbf{n}(\mathbf{w})$



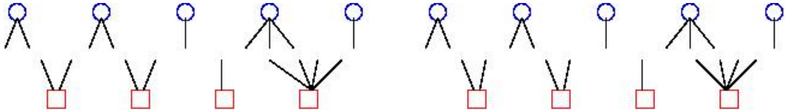
$$\mathbf{p} = (2, 2, 1, 0)$$

$$\mathbf{q} = (1, 2, 0, 1)$$

$$m = \sum_{d=1}^{L} dp_d = \sum_{w=1}^{L} wq_w \text{ (number of edges)}$$

Random hypergraphs

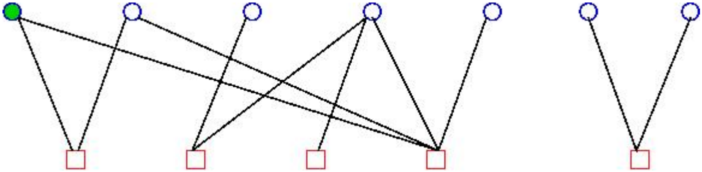
- $\mathbf{p} = (p_1, ..., p_L)$ and $\mathbf{q} = (q_1, ..., q_L)$ fixed vectors such that $\sum_{d=1}^{L} dp_d = \sum_{w=1}^{L} wq_w = m$
- Choose V, E finite sets, $\mathbf{d}$ degree function and $\mathbf{w}$ weight function such that $\mathbf{n}(\mathbf{d}) = N\mathbf{p}$ and $\mathbf{n}(\mathbf{w}) = N\mathbf{q}$, for $N \in \mathbb{N}^*$
- $G(\mathbf{d}, \mathbf{w})$ = set of all hypergraphs on $V \times E$ with degree function $\mathbf{d}$ and weight function $\mathbf{w}$
- Γ random hypergraph taken uniformly at random in $G(\mathbf{d}, \mathbf{w})$ ($\Gamma \sim U(\mathbf{d}, \mathbf{w})$)
- Number of edges $= Nm$, number of vertices $= N\|p\|_1$, number of hyper-edges $= N\|q\|_1$



Size of the largest component when $N \to \infty$?

Connected component of a given vertex :

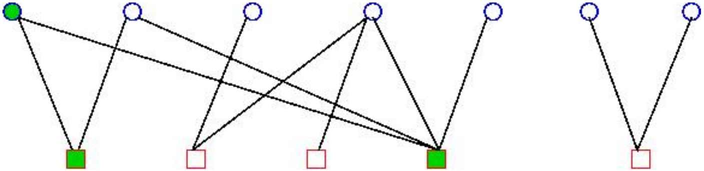
EXPLORATION ALGORITHM



Size of the largest component when $N \to \infty$?

Connected component of a given vertex :

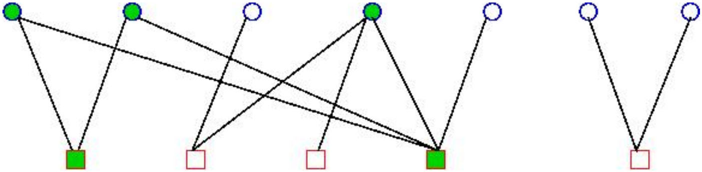
EXPLORATION ALGORITHM



Size of the largest component when $N \to \infty$?

Connected component of a given vertex :

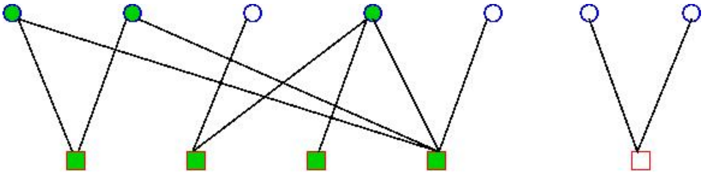
EXPLORATION ALGORITHM



Size of the largest component when $N \to \infty$?

Connected component of a given vertex :

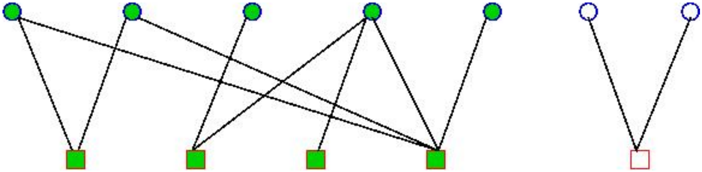
EXPLORATION ALGORITHM



Size of the largest component when $N \to \infty$?

Connected component of a given vertex :

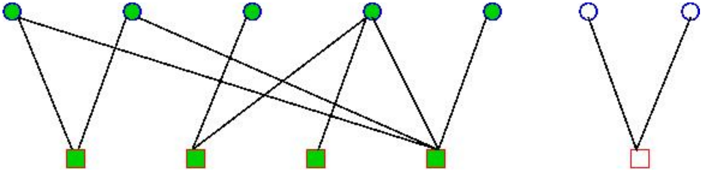
EXPLORATION ALGORITHM



Size of the largest component when $N \to \infty$?

Connected component of a given vertex :

EXPLORATION ALGORITHM



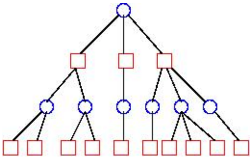
Need to explore a proportion α of the vertices

Branching process approximation : a way to guess the result

$\Gamma \sim U(\mathbf{d}, \mathbf{w})$ converges locally, when $N \to \infty$, to a tree

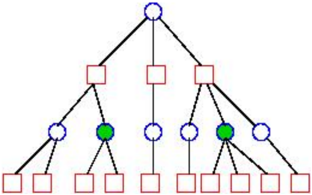
Corresponding random tree :

- Alternating one : generation of nodes of type V / generation of nodes of type E
- Except root, each node of type V has $d-1$ offsprings with probability $\frac{dp_d}{m}$
- Each of type E has $w-1$ offsprings with probability $\frac{wq_w}{m}$



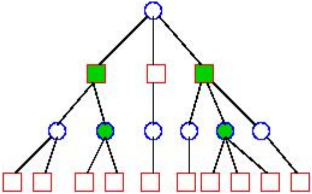
Algorithm on the tree

- Step 0 : Let $\alpha > 0$ and turn each vertex into **alive** with probability α
- Step 1a : turn into **alive** all individuals of type E having some alive vertex as an offspring
- Step 1b : turn into **alive** all individuals of type V having some alive hyper-edge as an offspring
- Repeat step 1 infinitely often



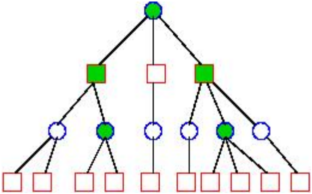
Algorithm on the tree

- Step 0 : Let $\alpha > 0$ and turn each vertex into **alive** with probability α
- Step 1a : turn into **alive** all individuals of type E having some alive vertex as an offspring
- Step 1b : turn into **alive** all individuals of type V having some alive hyper-edge as an offspring
- Repeat step 1 infinitely often



Algorithm on the tree

- Step 0 : Let $\alpha > 0$ and turn each vertex into **alive** with probability α
- Step 1a : turn into **alive** all individuals of type E having some alive vertex as an offspring
- Step 1b : turn into **alive** all individuals of type V having some alive hyper-edge as an offspring
- Repeat step 1 infinitely often



How to guess the result

Definitions

$$s_0 = g_0 = 1$$

$$s_n = \mathbb{P}(\text{after } n \text{ steps, a hyper-edge } e \text{ is not alive})$$

$$g_{n+1} = \mathbb{P}(\text{after } n \text{ steps, a vertex } v \text{ is not alive})$$

$$\begin{aligned} s_n &= \sum_w \frac{wq_w}{m}(g_n)^{w-1} \\ &=: \sigma(g_n) \\ g_{n+1} &= (1-\alpha)\sum_d \frac{dp_d}{m}(s_n)^{d-1} \\ &=: \phi_\alpha(g_n) \end{aligned}$$

How to guess the result

- ϕ_α maps continuously $[0,1]$ to $[0,1)$ and is increasing, so $(g_n)_{n\geq 0}$ converges to

$$z^*_\alpha = \text{largest root of } \phi_\alpha(z) = z \text{ in } [0,1)$$

- Proportion of alive vertices :

$$P(\alpha) = 1 - (1-\alpha)\sum_d \frac{p_d}{\|p\|_1}\left(\sigma(z^*_\alpha)\right)^d$$

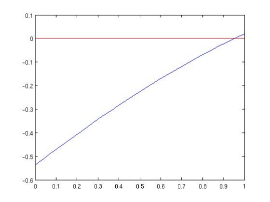
Two different behaviours

$$f_\alpha(z) = z - \phi_\alpha(z)$$

$$P(\alpha) \underset{\alpha \to 0}{\longrightarrow} 0$$

$$\Updownarrow$$

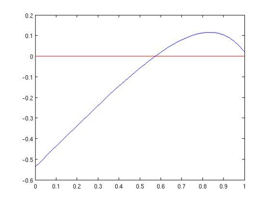
$$\lim_{\alpha \to 0} f'_\alpha(1) \geq 0$$



$$P(\alpha) \underset{\alpha \to 0}{\longrightarrow} \lambda_0 > 0$$

$$\Updownarrow$$

$$\lim_{\alpha \to 0} f'_\alpha(1) < 0$$



- D random variable such that $\mathbb{P}(D = d) \propto p_d$

$$\phi_D(z) = \sum_{d=1}^{L} \frac{p_d}{\|p\|_1} z^d$$

- W random variable such that $\mathbb{P}(W = w) \propto q_w$

$$\phi_W(z) = \sum_{w=1}^{L} \frac{q_w}{\|q\|_1} z^w$$

$$f_\alpha(z) = z - (1 - \alpha)\frac{1}{\mathbb{E}[D]}\phi'_D\left(\frac{1}{\mathbb{E}[W]}\phi'_W(z)\right)$$

When $N \to \infty$, is there a giant component of order N?

- $C_1(N)$ = largest connected component of $\Gamma \sim U(\mathbf{d}, \mathbf{w})$
- D random variable such that $\mathbb{P}(D = d) \propto p_d$
- W random variable such that $\mathbb{P}(W = w) \propto q_w$

Theorem

Case (i) *If* $\mathbb{E}[D(D-1)]\,\mathbb{E}[W(W-1)] \leq \mathbb{E}[D]\,\mathbb{E}[W]$
then for each $\epsilon > 0$, $\mathbb{P}\left(\frac{|C_1(N)|}{N} > \epsilon\right) \underset{N\to\infty}{\longrightarrow} 0$
(there is no giant component)

Case (ii) *If* $\mathbb{E}[D(D-1)]\,\mathbb{E}[W(W-1)] > \mathbb{E}[D]\,\mathbb{E}[W]$
then, there exists $\lambda > 0$ *such that, for each* $\epsilon > 0$,
$\mathbb{P}\left(\left|\frac{|C_1(N)|}{N} - \lambda\right| > \epsilon\right) \underset{N\to\infty}{\longrightarrow} 0$
(there exists a giant component of order N*)*

Exploring process

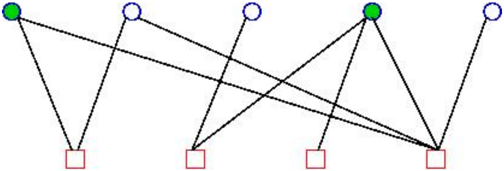
- Exploring the component of a given vertex
- Active vertices = those we want to explore the component
- $\alpha > 0$: activate each vertex independently with proba α
- 3 types of vertices : **sleeping**, **alive**, **dead**
  - **sleeping** = we haven't explored it
  - **alive** = we must explore it
  - **dead** = we have explored it

Exploring process : algorithm

1. Initially, label active vertices as alive, and non-active ones as sleeping
2. While there is a vertex that is alive do
3. Choose a vertex v uniformly at random among all alive vertices
4. For all hyper-edges e that contains v but no dead vertex do
5. For all sleeping vertices u connected with e do
6. Label u as alive
7. Label v as dead

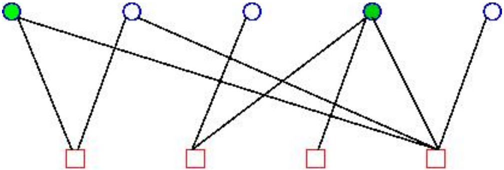
Exploring process : algorithm

1. Initially, label active vertices as alive, and non-active ones as sleeping
2. While there is a vertex that is alive do
3. Choose a vertex v uniformly at random among all alive vertices
4. For all hyper-edges e that contains v but no dead vertex do
5. For all sleeping vertices u connected with e do
6. Label u as alive
7. Label v as dead



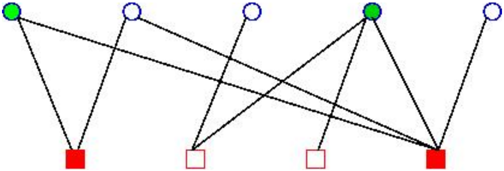
Exploring process : algorithm

1. Initially, label active vertices as alive, and non-active ones as sleeping
2. While there is a vertex that is alive do
3. Choose a vertex v uniformly at random among all alive vertices
4. For all hyper-edges e that contains v but no dead vertex do
5. For all sleeping vertices u connected with e do
6. Label u as alive
7. Label v as dead



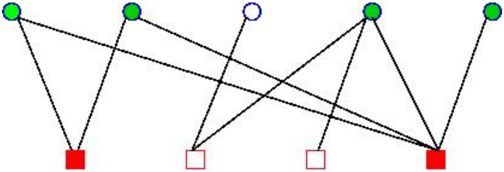
Exploring process : algorithm

1. Initially, label active vertices as alive, and non-active ones as sleeping
2. While there is a vertex that is alive do
3. Choose a vertex v uniformly at random among all alive vertices
4. For all hyper-edges e that contains v but no dead vertex do
5. For all sleeping vertices u connected with e do
6. Label u as alive
7. Label v as dead



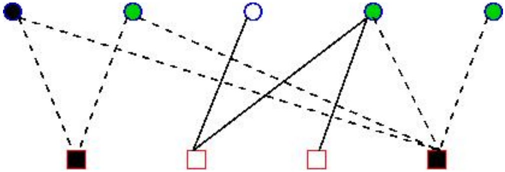
Exploring process : algorithm

1. Initially, label active vertices as alive, and non-active ones as sleeping
2. While there is a vertex that is alive do
3. Choose a vertex v uniformly at random among all alive vertices
4. For all hyper-edges e that contains v but no dead vertex do
5. For all sleeping vertices u connected with e do
6. Label u as alive
7. Label v as dead



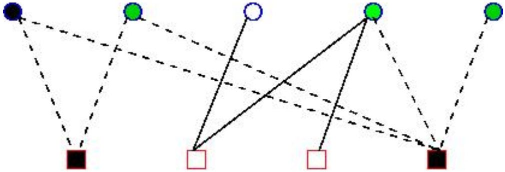
Exploring process : algorithm

1. Initially, label active vertices as alive, and non-active ones as sleeping
2. While there is a vertex that is alive do
3. Choose a vertex v uniformly at random among all alive vertices
4. For all hyper-edges e that contains v but no dead vertex do
5. For all sleeping vertices u connected with e do
6. Label u as alive
7. Label v as dead



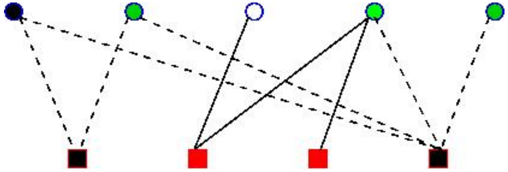
Exploring process : algorithm

1. Initially, label active vertices as alive, and non-active ones as sleeping
2. While there is a vertex that is alive do
3. Choose a vertex v uniformly at random among all alive vertices
4. For all hyper-edges e that contains v but no dead vertex do
5. For all sleeping vertices u connected with e do
6. Label u as alive
7. Label v as dead



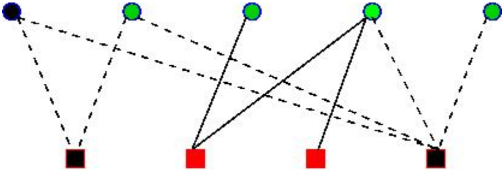
Exploring process : algorithm

1. Initially, label active vertices as alive, and non-active ones as sleeping
2. While there is a vertex that is alive do
3. Choose a vertex v uniformly at random among all alive vertices
4. For all hyper-edges e that contains v but no dead vertex do
5. For all sleeping vertices u connected with e do
6. Label u as alive
7. Label v as dead



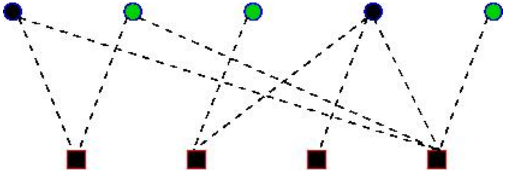
Exploring process : algorithm

1. Initially, label active vertices as alive, and non-active ones as sleeping
2. While there is a vertex that is alive do
3. Choose a vertex v uniformly at random among all alive vertices
4. For all hyper-edges e that contains v but no dead vertex do
5. For all sleeping vertices u connected with e do
6. Label u as alive
7. Label v as dead



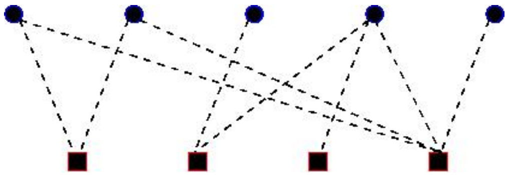
Exploring process : algorithm

1. Initially, label active vertices as alive, and non-active ones as sleeping
2. While there is a vertex that is alive do
3. Choose a vertex v uniformly at random among all alive vertices
4. For all hyper-edges e that contains v but no dead vertex do
5. For all sleeping vertices u connected with e do
6. Label u as alive
7. Label v as dead



Exploring process : algorithm

1. Initially, label active vertices as alive, and non-active ones as sleeping
2. While there is a vertex that is alive do
3. Choose a vertex v uniformly at random among all alive vertices
4. For all hyper-edges e that contains v but no dead vertex do
5. For all sleeping vertices u connected with e do
6. Label u as alive
7. Label v as dead



Jumping chain

- **Sequence of random hypergraphs**
  - $\Gamma_0 = \Gamma \sim U(\mathbf{d}, \mathbf{w})$
  - $\Gamma_n = \Gamma_{n-1} \backslash \{\text{dead vertex and its hyper-edges}\}$
  - $\mathbf{D}_n, \mathbf{W}_n$ = degree and weight functions of Γ_n
  - Conditionally on the past, $\Gamma_n \sim U(\mathbf{D}_n, \mathbf{W}_n)$
- Markov chain

$$\begin{cases} \xi_n^{\mathbf{d},\mathbf{d'},0} = \text{nb of } \textbf{non active} \text{ vertices with current degree } \mathbf{d} \text{ and initial degree } \mathbf{d'} \\ \xi_n^{\mathbf{d},\mathbf{d'},1} = \text{nb of } \textbf{active} \text{ vertices with current degree } \mathbf{d} \text{ and initial degree } \mathbf{d'} \\ \xi_n^{\mathbf{w}} = \text{nb of hyper-edges with current weight } \mathbf{w} \end{cases}$$

  - $\xi_n = \left(\xi_n^{d,d',k}, \xi_n^{w} : 0 \leq d \leq d',\ k \in \{0,1\},\ 0 \leq w\right)$
  - $(\xi_n)_{n\geq 0}$ **is a Markov chain**

Differential equation approximation

- (X_t) continuous-time Markov chain with jump chain (ξ_n)
- Coordinate functions :

$$Y_t = \left(\frac{X_t^{d,d,0}}{N}, \frac{X_t^w}{N} : 1 \leq d \leq L, 1 \leq w \leq L \right)$$

- Estimation of the generator
- Differential equation approximation :

$$\left\{ \begin{array}{lcl} x_t^w & = & e^{-tw} q_w \\ x_t^{d,d,0} & = & \sigma(e^{-t})^d \, (1-\alpha) p_d \end{array} \right.$$

- Terminal values

Proportion of dead vertices

- End of the algorithm

$\iff$ number of edges connected with alive vertices $= 0$
$\iff \sum_w w x_t^w - \sum_d d x_t^{d,d,0} = 0$
$\iff f_\alpha(e^{-t}) = 0$

- Proportion of dead vertices :

$$P(\alpha) = 1 - \frac{|\{\text{remained vertices}\}|}{|\{\text{initial vertices}\}|} = 1 - \frac{1}{\|p\|_1} \sum_d \sigma(z_\alpha^*)^d (1-\alpha) p_d$$

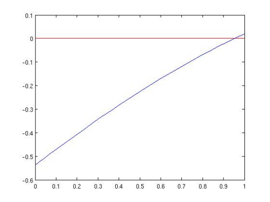
Two different behaviours

$$f_\alpha(z) = z - \phi_\alpha(z)$$

$$P(\alpha) \underset{\alpha \to 0}{\longrightarrow} 0$$

$$\Updownarrow$$

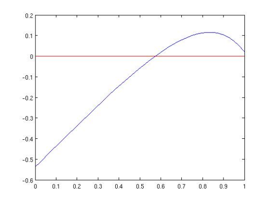
$$\lim_{\alpha \to 0} f'_\alpha(1) \geq 0$$



$$P(\alpha) \underset{\alpha \to 0}{\longrightarrow} \lambda_0 > 0$$

$$\Updownarrow$$

$$\lim_{\alpha \to 0} f'_\alpha(1) < 0$$



Conclusion

- As for random graphs, existence of a phase transition for the appearance of the giant component

- Branching process : a way to guess the result
- Markov chain : a tool for proving it

Conclusion

- As for random graphs, existence of a phase transition for the appearance of the giant component
- Branching process : a way to guess the result
- Markov chain : a tool for proving it

Thanks for your attention !

Some bibliography

[1] Bollobas, *Random Graphs*, Cambridge Univ. Press, 2001 (1st publication : Academic Press, 1985)

[2] Coja-Oghlan, Moore & Sanwalani, *Counting Connected Graphs and Hypergraphs via the Probabilistic Method*, Rand. Struct. and Algorithms, 2007

[3] Darling & Norris, *Differential Equation Approximations for Markov Chains*, Prob. Surveys, 2008

[4] Janson & Luczak, *A New Approach to the Giant Component Problem*, Rand. Struct. and Algorithms, 2009

[5] Lelarge, *Diffusion and Cascading Behaviour on Clustered Networks* (not published)

How to guess the result

Definitions

$$\begin{aligned} s_0 &= g_0 = 1 \\ s_n &= \mathbb{P}(\text{after } n \text{ steps, a hyper-edge } e \text{ isn't alive}) \\ g_{n+1} &= \mathbb{P}(\text{after } n \text{ steps, a vertex } v \text{ isn't alive}) \\ g_{n+1}^d &= \mathbb{P}(\text{a vertex of degree } d \text{ isn't alive at time } n) \end{aligned}$$

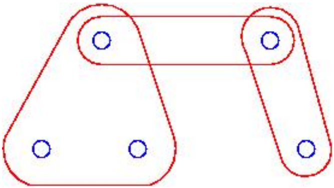
$$\begin{aligned} s_n &= \mathbb{P}(\text{every offspring of } e \text{ wasn't alive at time } n-1) \\ &= \sum_w \mathbb{P}(e \text{ has } w-1 \text{ offsprings}) \left(\mathbb{P}(\text{a given offspring } v \text{ isn't alive at time } n-1)\right)^{w-1} \\ &= \sum_w \frac{w q_w}{m} (g_n)^{w-1} \\ &=: \sigma(g_n) \end{aligned}$$

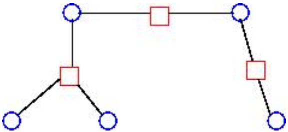
How to guess the result

$$
\begin{aligned}
g^d_{n+1} &= \mathbb{P}(\text{a vertex } v \text{ of degree } d \text{ isn't alive at time } n \,|\, v \in V_A)\mathbb{P}(v \in V_A) \\
&\quad + \mathbb{P}(\text{a vertex } v \text{ of degree } d \text{ isn't alive at time } n \,|\, v \notin V_A)\mathbb{P}(v \notin V_A) \\
&= (1-\alpha)\, \mathbb{P}(\text{a vertex } v \text{ of degree } d \text{ isn't alive at time } n \,|\, v \notin V_A) \\
&= (1-\alpha)\, \mathbb{P}(\text{none of the } d-1 \text{ offsprings of } v \text{ is alive at time } n) \\
&= (1-\alpha)\, s_n^{d-1} \\
&= (1-\alpha)\, (\sigma(g_n))^{d-1}
\end{aligned}
$$

$$
\begin{aligned}
g_{n+1} &= \sum_d \mathbb{P}(v \text{ isn't alive at time } n \,|\, v \text{ has degree } d)\mathbb{P}(v \text{ has degree } d) \\
&= (1-\alpha) \sum_d \frac{dp_d}{m} (\sigma(g_n))^{d-1} \\
&=: \phi_\alpha(g_n)
\end{aligned}
$$

Graphs with clustering





Upper bound : idea

For each $\alpha > 0$:

- $P_N(\alpha)$ = proportion of alive vertices at the end of the algorithm
- $P(\alpha) = \lim_{N \to \infty} P_N(\alpha)$ (limit in probability)
- If we activate some vertex in $C_1(N)$, then

$$\frac{|C_1(N)|}{Nn} \leq P_N(\alpha)$$

- The probability of activating no vertex in $C_1(N)$ tends to 0 (when $N \to \infty$)