interne Groupe

Tools for Community Detection

Jérôme Galtier, Orange Labs

Mascotte seminar, March 12, 2009

Image by Sarah Klan of the Young Creatives Network. The future's bright.

orange™

the search engines: a gateway to the web?

www.searchenginegenie.com

Share of searches	Jul 2007	May 2008
Google	55.2%	61.6%
Yahoo	23.5%	20.4%
Microsoft sites	12.3%	9.1%
AOL LLC	4.4%	4.6%
Ask network	4.7%	4.3%

in the US, (source: comScore qSearch)

A massive set of data

Search engine	Reported size (mid 2005)
Google	11.3 billion
MSN	5.0 billion
Yahoo	19.2 billion
Ask Jeeves	2.5 billion
	in number of pages

our approach

$\longrightarrow$ employ our **graph theory** background

our approach

→ employ our **graph theory** background

→ give **alternative approaches** to search the web

our approach

$\longrightarrow$ employ our **graph theory** background

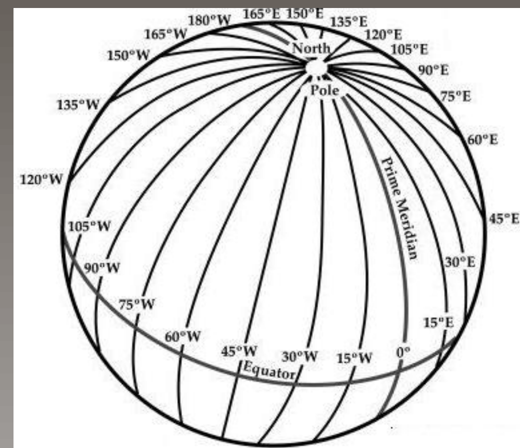
$\longrightarrow$ give **alternative approaches** to search the web

the PACK 237: cartographie Pages Web

- brevet 04843 (A. Laugier) : randonnée
- brevet 05618 (A. Laugier, S. Raymond) : mineur
- brevet 05711 (J. Galtier) : webworld
- brevet 06448 (J. Galtier, A. Skoda, J. Fonlupt) : strength of graphs

$\longrightarrow$ integrated in the PAC by PIV

idea of WEBWORLD

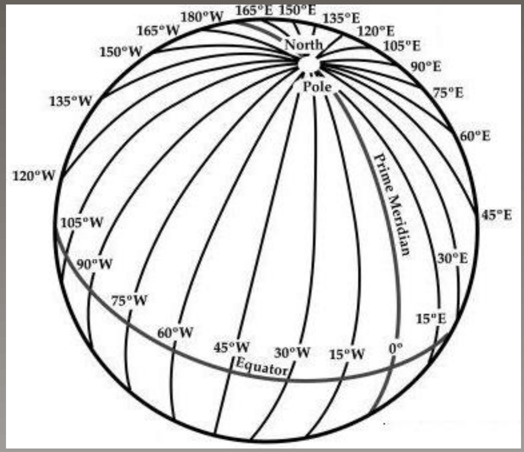


WWW $\mapsto$

URL $\longrightarrow$ a point in the unit sphere

idea of WEBWORLD

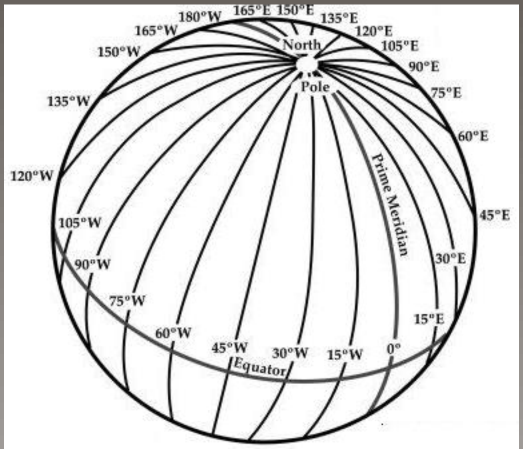
WWW $\mapsto$



URL $\longrightarrow$ a point in the unit sphere

the documents are viewed as **particules**.

idea of WEBWORLD

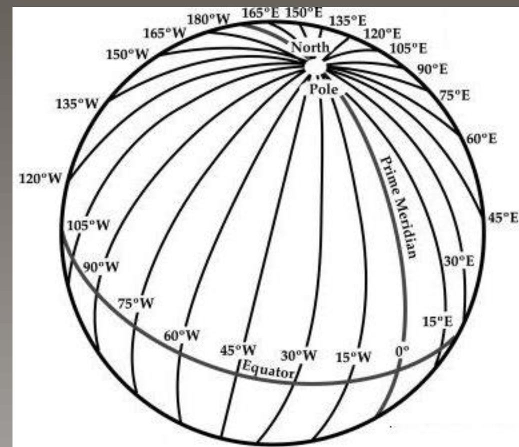


$$\begin{array}{ccl} \text{WWW} & \mapsto & \\ \text{URL} & \longrightarrow & \text{a point in the unit sphere} \end{array}$$

the documents are viewed as **particules**.

2 particules attract each other if and only if theu are connected.

idea of WEBWORLD



WWW $\mapsto$

URL $\longrightarrow$ a point in the unit sphere

the documents are viewed as **particules**.

2 particules attract each other if and only if theu are connected. otherwise, they repel each other.

the mathematics inside webworld

we are given an undirected graph $G = (V, E)$.

each edge $\{u, v\}$ is associated to a weight $w_{u,v} \in [0, 1]$, that emphasizes the **degree of connection between the pages**.

the mathematics inside webworld

we are given an undirected graph $G = (V, E)$.

each edge $\{u, v\}$ is associated to a weight $w_{u,v} \in [0, 1]$, that emphasizes the **degree of connection between the pages**.

we set δ as follows: $\delta_{u,v} = \left| \begin{array}{ll} 1 & \text{if } \{u, v\} \notin E \\ 1 - w_{u,v} & \text{if } \{u, v\} \in E. \end{array} \right.$

the mathematics inside webworld

we are given an undirected graph $G = (V, E)$.

each edge $\{u, v\}$ is associated to a weight $w_{u,v} \in [0, 1]$, that emphasizes the **degree of connection between the pages**.

we set δ as follows: $\delta_{u,v} = \left| \begin{array}{ll} 1 & \text{if } \{u,v\} \notin E \\ 1 - w_{u,v} & \text{if } \{u,v\} \in E. \end{array} \right.$

we associate to each vertex u a point $X(u)$ in the 3D unit sphere.

the mathematics inside webworld

we are given an undirected graph $G = (V, E)$.

each edge $\{u, v\}$ is associated to a weight $w_{u,v} \in [0, 1]$, that emphasizes the **degree of connection between the pages**.

we set δ as follows: $\delta_{u,v} = \left| \begin{array}{ll} 1 & \text{if } \{u, v\} \notin E \\ 1 - w_{u,v} & \text{if } \{u, v\} \in E. \end{array} \right.$

we associate to each vertex u a point $X(u)$ in the 3D unit sphere.

problem: $\boxed{\text{maximize} \sum_{\{u,v\} \in E} \delta_{u,v} ||X(u) - X(v)||^2.}$

principle of the algorithm of Webworld

the algorithm used is **incremental**.

principle of the algorithm of Webworld

the algorithm used is **incremental**.

Each point of the sphere (corresponding to a document) is placed at its unique best position with respect to the others

principle of the algorithm of Webworld

the algorithm used is **incremental**.

Each point of the sphere (corresponding to a document) is placed at its unique best position with respect to the others

The processus is iterated. . .

principle of the algorithm of Webworld

the algorithm used is **incremental**.

Each point of the sphere (corresponding to a document) is placed at its unique best position with respect to the others

The processus is iterated. . .

→ This procedure necessarily converges to an optimum. (no cyclic process)

principle of the algorithm of Webworld

the algorithm used is **incremental**.

Each point of the sphere (corresponding to a document) is placed at its unique best position with respect to the others

The processus is iterated. . .

$\rightarrow$ This procedure necessarily converges to an optimum. (no cyclic process)

in practice, **3 min** are required to update 97.10^6 **nodes**.

the algorithm of Webworld

the algorithm used is **incremental**.

the $\mathfrak{algorithm}$ of Webworld

the algorithm used is **incremental**.

we update each vertex u as follows:

the algorithm of Webworld

the algorithm used is **incremental**.

we update each vertex u as follows:

- compute the value $Y(u) := \sum_{v \in V - \{u\}} \delta_{u,v} X(v)$,

the algorithm of Webworld

the algorithm used is **incremental**.

we update each vertex u as follows:

- compute the value $Y(u) := \sum_{v \in V - \{u\}} \delta_{u,v} X(v)$,
- if $Y(u) \neq 0$, we perform the operation $X(u) := -Y(u)/||Y(u)||$,

the algorithm of Webworld

the algorithm used is **incremental**.

we update each vertex u as follows:

- compute the value $Y(u) := \sum_{v \in V - \{u\}} \delta_{u,v} X(v)$,
- if $Y(u) \neq 0$, we perform the operation $X(u) := -Y(u)/||Y(u)||$,
- if $Y(u) = 0$, choose $X(u)$ at random.

the algorithm of Webworld

the algorithm used is **incremental**.

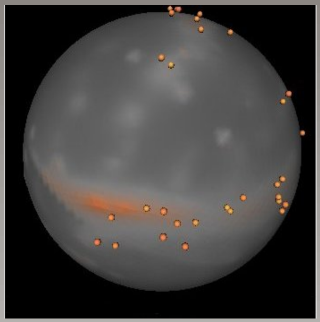
we update each vertex u as follows:

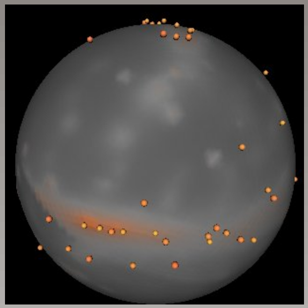
- compute the value $Y(u) := \sum_{v \in V - \{u\}} \delta_{u,v} X(v)$,
- if $Y(u) \neq 0$, we perform the operation $X(u) := -Y(u)/||Y(u)||$,
- if $Y(u) = 0$, choose $X(u)$ at random.

$\longrightarrow$ we increase the estimated **Shannon** capacity for the graph.

some results

the green parts of the planet represent the concentrated areas of sites, while the blue parts are the spase areas. the red bullets are the answers in voila.fr for the request.





request “orange” request “france telecom”

Strength of graphs

- Research started in 2003 with Alexandre Skoda PhD thesis.

Strength of graphs

- Research started in 2003 with Alexandre Skoda PhD thesis.
- Goal: find a mathematical approach to detect communities.

Strength of graphs

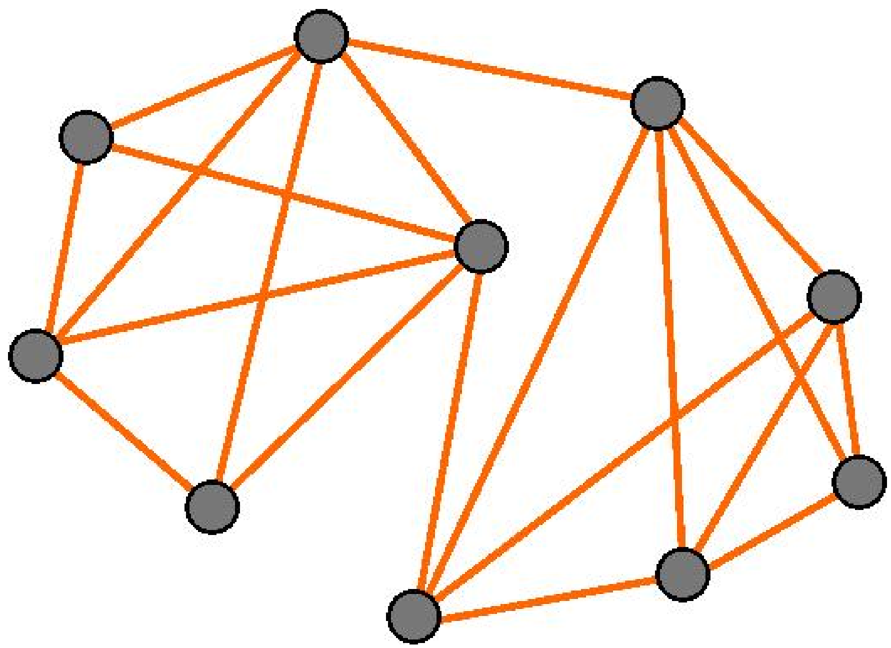
- Research started in 2003 with Alexandre Skoda PhD thesis.
- Goal: find a mathematical approach to detect communities.
- Software already written.

Strength of graphs

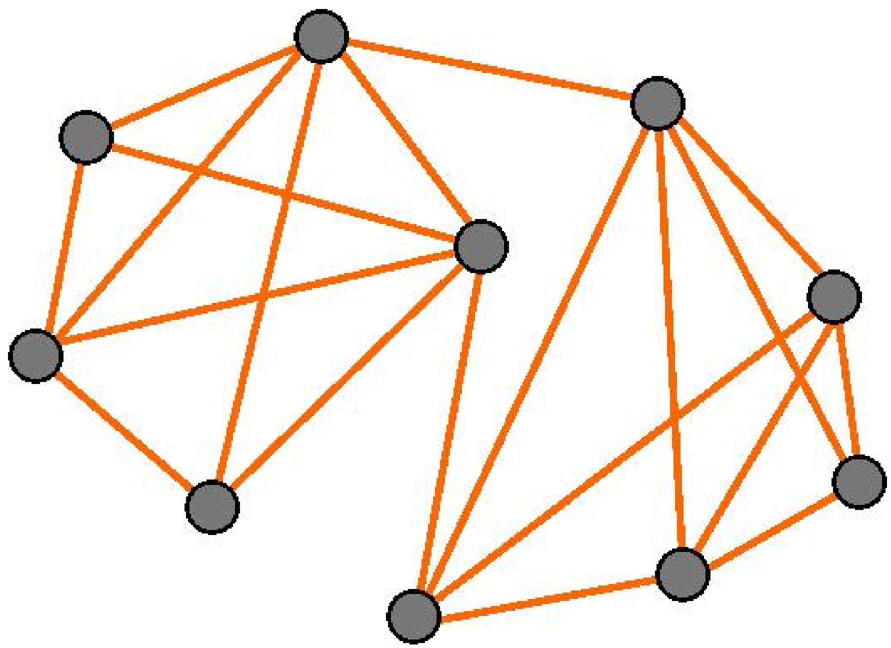
- Research started in 2003 with Alexandre Skoda PhD thesis.
- Goal: find a mathematical approach to detect communities.
- Software already written.

$\rightarrow$ it does not give the result of a heuristic but a 'photography' of properties of the set of documents. The more accurate, the more CPU time is required to compute it ($\times 2$ in precision $\Leftrightarrow$ $\times 4$ in computation).

Strength of graphs: intuition

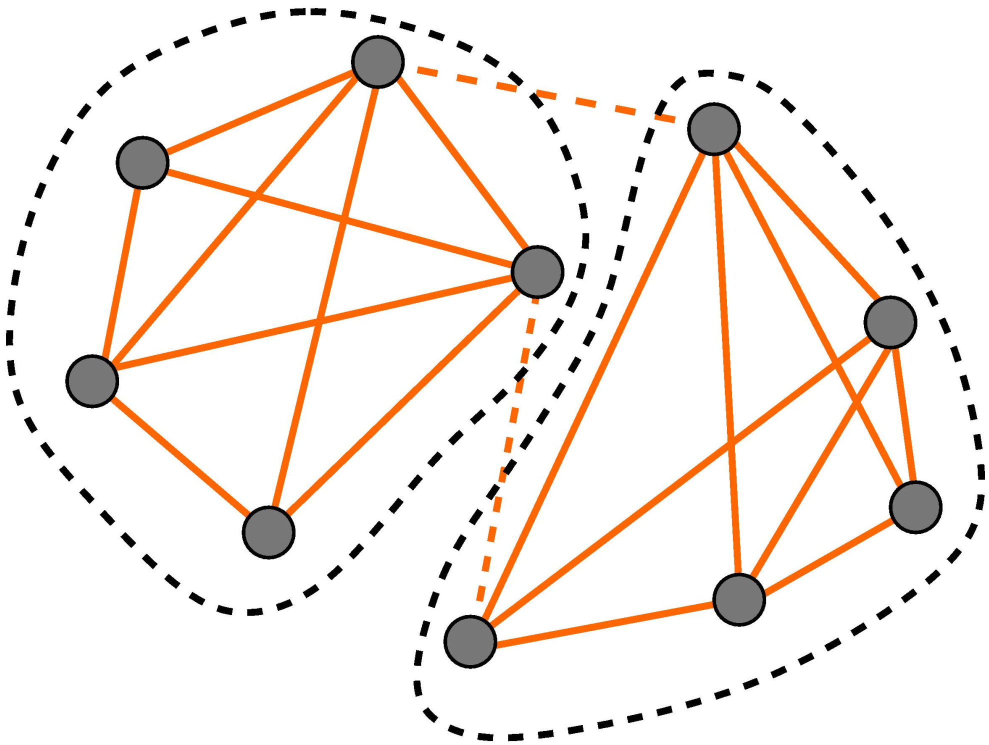


Strength of graphs: intuition



$\rightarrow$ minimize the strength that is the ratio $\frac{\text{edges withdrawn}}{\text{created components}}$.

Strength of graphs: intuition



→ each sub-community that is not a singleton is then redivided and has provably a better strength.

the Tutte Nash-Williams theorem (1961)

G contains k edge-disjoint spanning trees

$$\Leftrightarrow$$

$$\sigma(G) \geq k$$

a word on the bibliography

Strength of graph is linked to *graph partitionning* and serves as the underground algorithm to approximate the *minimum cut* of a graph in almost linear time.

Many algorithms use the maximum flow, which runs with best complexity $MF(n,m) = O(\min(\sqrt{m}, n^{2/3}) m \log(n^2/m + 2))$ (Goldberg & Rao, 1998).

1984	Cunningham	$O(nm\ MF(n,n^2))$	Exact
1988	Gabow &	$O(\sqrt{\frac{m}{n}}(m + n\log n)\log\frac{m}{n})$	Integer
	Westermann	$O(nm\log\frac{m}{n})$	Integer
1991	Gusfield	$O(n^3 m)$	Exact
1991	Plotkin et ali	$O(m\sigma(G)\log(n)^2/\varepsilon^2)$	Within $1+\varepsilon$
1993	Trubin	$O(n\ MF(n,m))$	Exact
1998	Garg & Konemann	$O(m^2\sigma(G)\log(n)/\varepsilon^2)$	Within $1+\varepsilon$
2008	Galtier	$O(m\log(n)^2/\varepsilon^2)$	Within $1+\varepsilon$

this presentation

- a first linear programming formulation of size polynomial in the size of the problem,

this presentation

- a first linear programming formulation of size polynomial in the size of the problem,
- sketch proof of the $1+\varepsilon$ approximation in time $O(m\log(n)^2/\varepsilon^2)$

an equivalence theorem

Let $\mathcal{T}$ be the set of all spanning trees of the graph G.

$$\sigma(G) = \max\left(\sum_{T\in\mathcal{T}} \lambda_T : \forall T \in \mathcal{T}\ \lambda_\mathrm{T} \geq 0 \text{ and } \forall \mathrm{e} \in \mathrm{E}\ \sum_{\mathrm{T}\ni\mathrm{e}} \lambda_\mathrm{T} \leq 1\right)$$

an equivalence theorem

Let $\mathcal{T}$ be the set of all spanning trees of the graph G.

$$\sigma(G) = \max\left(\sum_{T\in\mathcal{T}} \lambda_T : \forall T \in \mathcal{T}\ \lambda_\mathrm{T} \geq 0 \text{ and } \forall \mathrm{e} \in \mathrm{E}\ \sum_{\mathrm{T}\ni\mathrm{e}} \lambda_\mathrm{T} \leq 1\right)$$

By linear duality we can reformulate it as follows:

$$\sigma(G) = \min\left(\sum_{e\in E} y_e : \forall e \in E\ y_e \geq 0 \text{ and } \forall T \in \mathcal{T}\ \sum_{\mathrm{e}\in\mathrm{T}} \mathrm{y_e} \geq 1\right).$$

linearizing the problem. . .

Consider the set of $\mathbb{R}^E$ given by:

$$\mathcal{S} = \left\{ z \in \mathbb{R}^E \ : \ \exists T \in \mathcal{T} \ \forall e \in E \ z_e = \chi_{\{e \in T\}} \right\},$$

linearizing the problem. . .

Consider the set of $\mathbb{R}^E$ given by:

$$\mathcal{S} = \left\{ z \in \mathbb{R}^E \ : \ \exists T \in \mathcal{T} \ \forall e \in E \ z_e = \chi_{\{e \in T\}} \right\},$$

and note that $\exists A, b \quad conv(S) = \left\{ z \ : \ \exists f \quad A \cdot \begin{pmatrix} f \\ z \end{pmatrix} \leq b. \right\}$

linearizing the problem. . .

Consider the set of $\mathbb{R}^E$ given by:

$$\mathcal{S} = \left\{ z \in \mathbb{R}^E \ : \ \exists T \in \mathcal{T} \ \forall e \in E \ z_e = \chi_{\{e \in T\}} \right\},$$

and note that $\exists A, b \quad conv(S) = \left\{ z \ : \ \exists f \quad A \cdot \begin{pmatrix} f \\ z \end{pmatrix} \leq b. \right\}$

Now we can say:

$$\sigma(G) = \min \left(\sum_{e \in E} y_e : \forall e \in E, y_e \geq 0, \forall z \in \mathcal{S}, \sum_{\mathrm{e} \in \mathrm{E}} \mathrm{z_e y_e} \geq 1 \right),$$

pushing further the decomposition

(i) $$\sum_{e\in E} y_e z_e \geq 1 \qquad \forall z \in \mathcal{S},$$

(ii) $$\sum_{e\in E} y_e z_e \geq 1 \qquad \forall z \in conv(\mathcal{S}),$$

(iii) $$\sum_{e\in E} y_e z_e \geq 1 \qquad \forall (z, f) \text{ such that } A \cdot \begin{pmatrix} f \\ z \end{pmatrix} \leq b,$$

(iv) For all $\varepsilon > 0$, there are no solution for

$$\begin{cases} A \cdot \begin{pmatrix} f \\ z \end{pmatrix} & \leq & b \\ \sum z_e y_e & \leq & 1 - \varepsilon, \end{cases}$$

(v) For all $\varepsilon > 0$, there exists a $x \geq 0$ such that

$$\begin{cases} x^t \cdot A + y = 0 \\ x^t \cdot b + (1 - \varepsilon) < 0, \end{cases}$$

(vi) There exists a $x \geq 0$ such that

$$\begin{cases} x^t \cdot A + y = 0 \\ x^t \cdot b + 1 \leq 0. \end{cases}$$

linear formulation (Pick a “root” $r \in V$)

$$\sigma(G) = \min \sum_{e \in E} y_e$$

$$-\gamma_v^k + \gamma_w^k + \mu_{\overrightarrow{vw}}^k \geq 0, \qquad \forall \overrightarrow{vw} \in \vec{E}, \quad \forall k \in V - \{r\}$$

$$\varphi - \sum_{k \in V - \{r\}} \mu_{\vec{e}}^k + y_e \geq 0 \qquad \forall \vec{e} \in \vec{E}$$

$$-\sum_{k \in V - \{r\}} \gamma_r^k + \sum_{k \in V - \{r\}} \gamma_k^k + (n-1)\varphi \leq -1$$

$$\varphi \geq 0, \mu_{\vec{e}}^k \geq 0 \qquad \forall \vec{e} \in \vec{E}, \quad \forall k \in V - \{r\}.$$

(variables $y_e,\ e \in E$, $\gamma_v^k,\ v, k \in V$, $\mu_{\vec{e}}^k,\ k \in V, \vec{e} \in \vec{E}$, and φ)

A word on the linear approximation

The algorithm as basis takes a pushing flow scheme.

(0) Each edge $e \in E$ receives a very small weight $w(e) = \delta = O\left(\frac{1}{nm^{\frac{1}{4\varepsilon}}}\right)$,

A word on the linear approximation

The algorithm as basis takes a pushing flow scheme.

(0) Each edge $e \in E$ receives a very small weight $w(e) = \delta = O\left(\frac{1}{nm^{\frac{1}{4\varepsilon}}}\right)$,

(1) At each step, compute a minimum spanning tree T with respect to w,

A word on the linear approximation

The algorithm as basis takes a pushing flow scheme.

(0) Each edge $e \in E$ receives a very small weight $w(e) = \delta = O\left(\frac{1}{nm^{\frac{1}{4\varepsilon}}}\right)$,

(1) At each step, compute a minimum spanning tree T with respect to w,

(2) For each $e \in T$, update $w(e) := w(e) * (1 + \varepsilon)$,

A word on the linear approximation

The algorithm as basis takes a pushing flow scheme.

(0) Each edge $e \in E$ receives a very small weight $w(e) = \delta = O\left(\frac{1}{nm^{\frac{1}{4\varepsilon}}}\right)$,

(1) At each step, compute a minimum spanning tree T with respect to w,

(2) For each $e \in T$, update $w(e) := w(e) * (1 + \varepsilon)$,

(3) If $w(T) < 1$ go to (1),

A word on the linear approximation

The algorithm as basis takes a pushing flow scheme.

(0) Each edge $e \in E$ receives a very small weight $w(e) = \delta = O\left(\frac{1}{nm^{\frac{1}{4\varepsilon}}}\right)$,

(1) At each step, compute a minimum spanning tree T with respect to w,

(2) For each $e \in T$, update $w(e) := w(e) * (1 + \varepsilon)$,

(3) If $w(T) < 1$ go to (1),

(4) Output $\sum_{e \in E} w(e)$.

A word on the linear approximation

The algorithm as basis takes a pushing flow scheme.

(0) Each edge $e \in E$ receives a very small weight $w(e) = \delta = O\left(\frac{1}{nm^{\frac{1}{4\varepsilon}}}\right)$,

(1) At each step, compute a minimum spanning tree T with respect to w,

(2) For each $e \in T$, update $w(e) := w(e) * (1 + \varepsilon)$,

(3) If $w(T) < 1$ go to (1),

(4) Output $\sum_{e \in E} w(e)$.

$\rightarrow$ this is an-$(1 + \varepsilon)$ approximation

(Plotkin, Shmoys, Tardos 1991, Young 1995).

Brute analysis of the complexity

Each edge cannot be updated more that $\frac{\log(\delta)}{\log(1+\varepsilon)} = O(\frac{\log(n)}{\varepsilon^2})$,

Each step updates $n-1$ edges and runs in $O(m\log(n))$,

$\rightarrow$ the computation takes less than $O(\frac{m^2\log(n)^2}{n\varepsilon^2})$.

Brute analysis of the complexity

Each edge cannot be updated more that $\frac{\log(\delta)}{\log(1+\varepsilon)} = O(\frac{\log(n)}{\varepsilon^2})$,

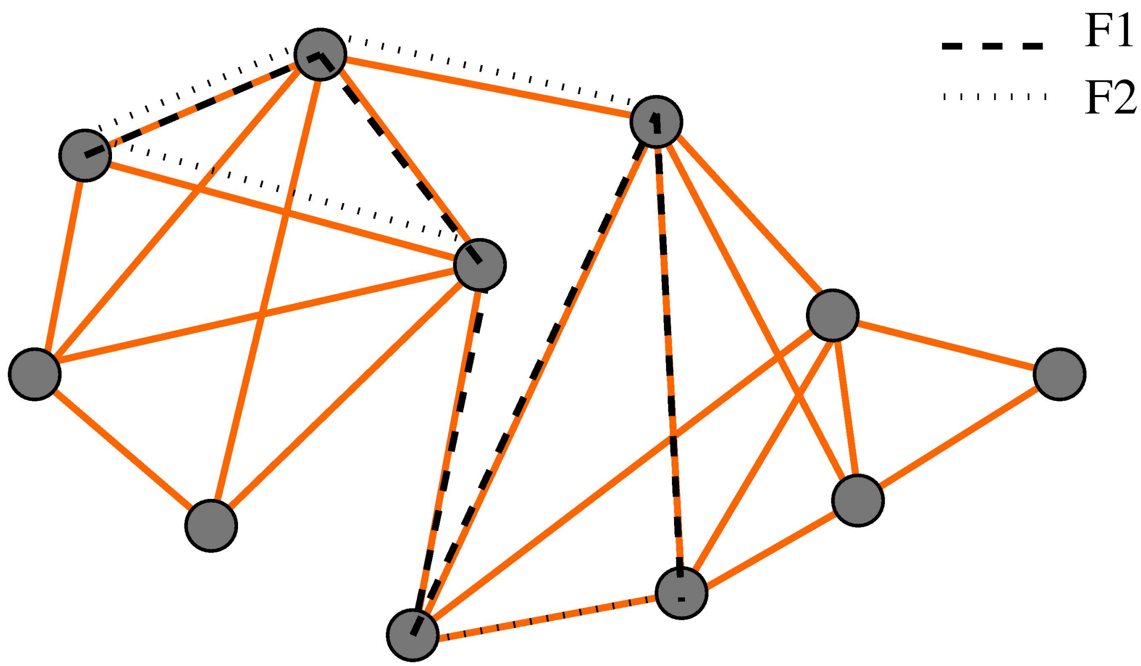
Each step updates $n-1$ edges and runs in $O(m \log(n))$,

$\rightarrow$ the computation takes less than $O(\frac{m^2 \log(n)^2}{n\varepsilon^2})$.

how can we gain the factor m/n ???

order on forests

A forest F_1 is more connecting than a forest F_2 ($F_1 \succeq F_2$) if the endpoints of any path of F_2 are connected in F_1.



augment and connecting order

Let $e \in E$. We say that e is independent of forest F is there is no path in F between endpoints of e. Otherwise it is dependent.

Augmenting F by an independent edge e to F : $F := F \cup \{e\}$.

Remark: Suppose $F_1 \succeq F_2$ and e is independent of F_1, then e is independent of F_2.

edge addition on ordered forests

idea: order the forests to add edges

$F_1 \succeq F_2 \succeq \cdots \succeq F_p$

take $e \in E$.

augment the first F_i such that e is independent to F_i.

edge addition on ordered forests

idea: order the forests to add edges

$F_1 \succeq F_2 \succeq \cdots \succeq F_p$

take $e \in E$.

augment the first F_i such that e is independent to F_i.

$\rightarrow$ a tree will be built in $O(n \log(n) \log(p))$ in average.

edge addition on ordered forests

idea: order the forests to add edges

$F_1 \succeq F_2 \succeq \cdots \succeq F_p$

take $e \in E$.

augment the first F_i such that e is independent to F_i.

$\longrightarrow$ a tree will be built in $O(n \log(n) \log(p))$ in average.

$\longrightarrow$ this works also with weighted edges in increasing order.

edge addition on ordered forests

idea: order the forests to add edges

$F_1 \succeq F_2 \succeq \cdots \succeq F_p$

take $e \in E$.

augment the first F_i such that e is independent to F_i.

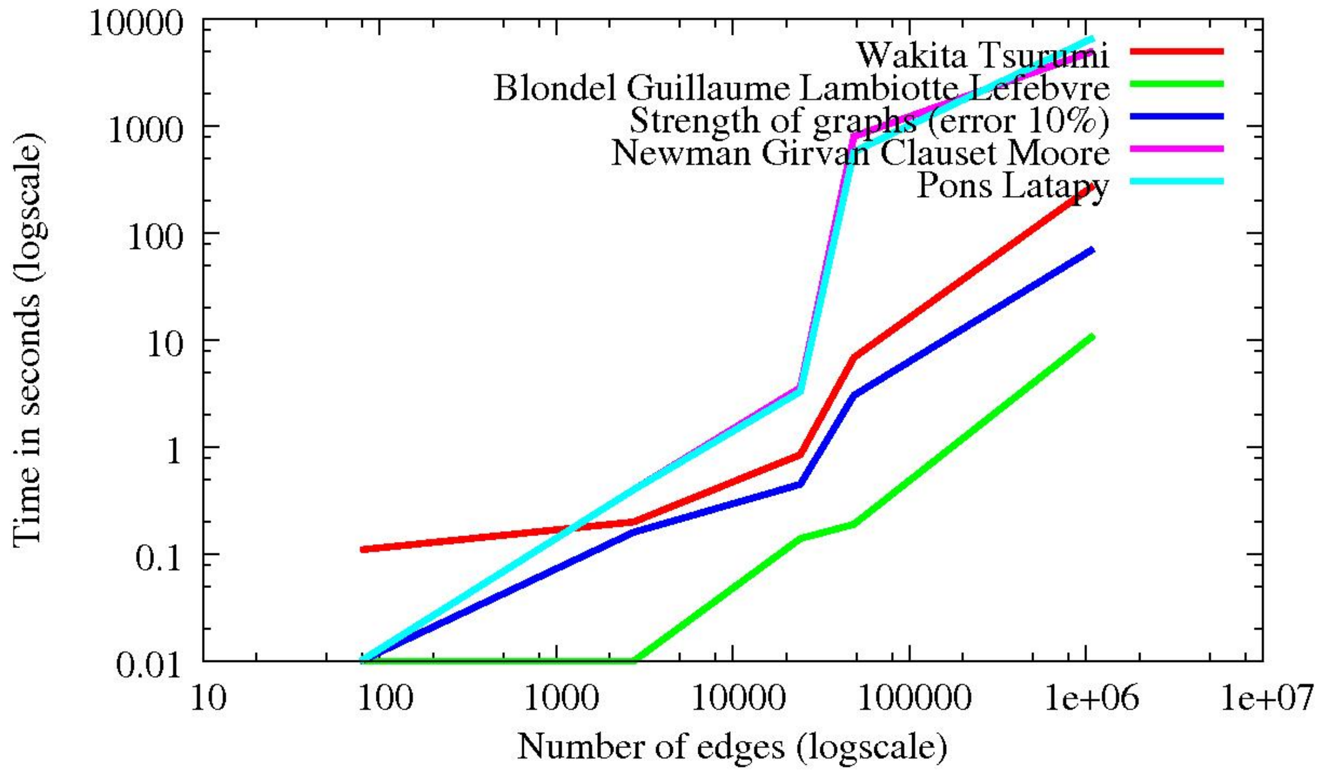
$\longrightarrow$ a tree will be built in $O(n \log(n) \log(p))$ in average.

$\longrightarrow$ this works also with weighted edges in increasing order.

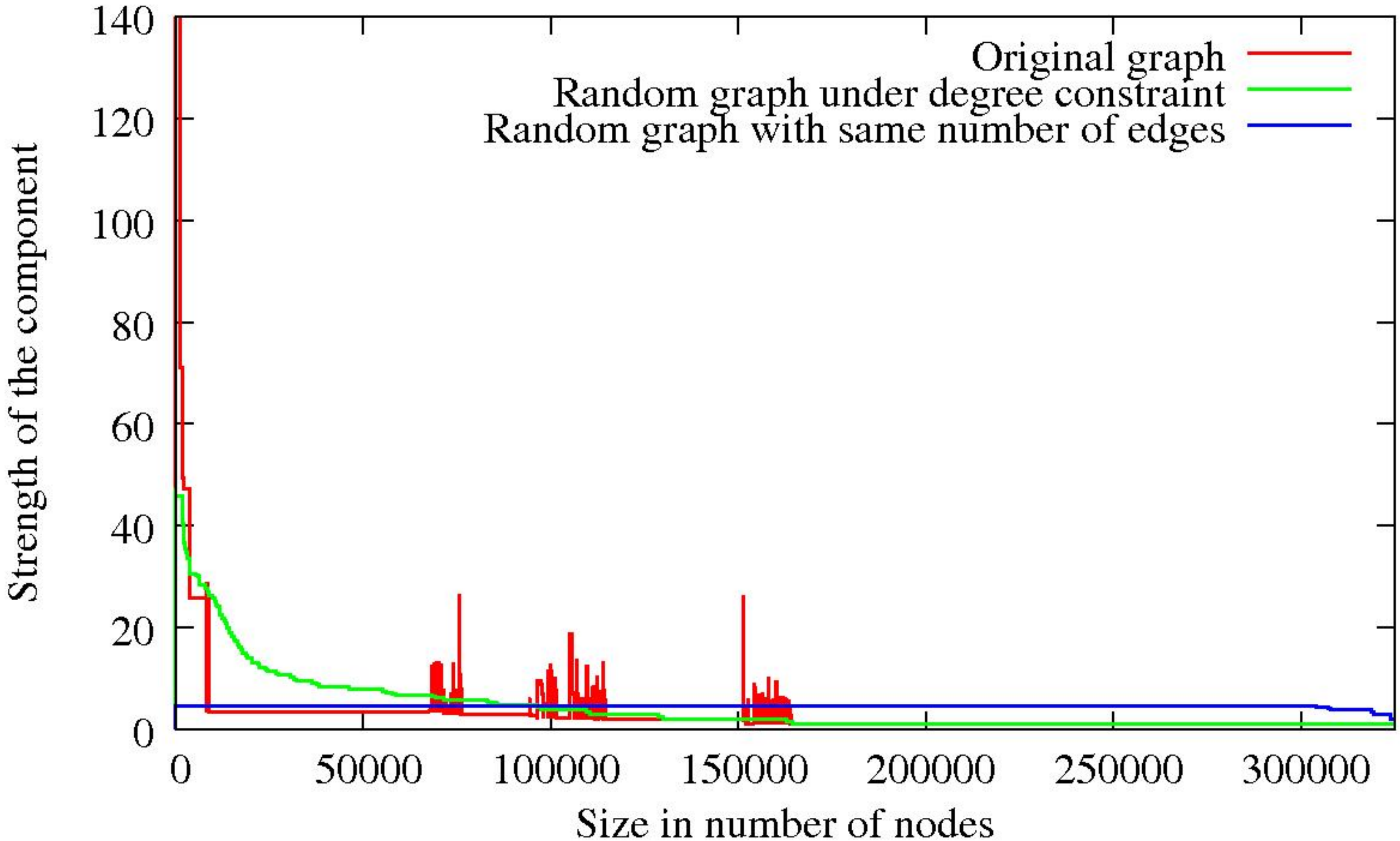
this gives an $O(m \log(n)^3/\varepsilon^2)$ algorithm.

Computational linearity

The algorithm is almost linear with the number of links between documents. Here compared with popular heuristics and datasets:



Graph spectrum (web of Albert, Jeong, Barabasi)



Achievements

- Webworld is an efficient heuristic that gives a particule representation of data on a sphere. It has been tested successfully with networks of 240 millions of nodes and 640 millions of links.

- the strength of graph gives a photography of the connectivity of a network that can be computed in almost linear time. It has been tested on networks with as much as 326 000 nodes and 1.5 million of links (around one minute of computation for 10% precision).

What next?

♣ Research questions: fiedler vector, modularity, k-densest subgraph. . .

What next?

♣ Research questions: fiedler vector, modularity, k-densest subgraph. . .

♣ Worl with Orange portail + orange labs

What next?

- ♣ Research questions: fiedler vector, modularity, k-densest subgraph. . .
- ♣ Worl with Orange portail + orange labs
- ♣ New ANR started

Thanks for your attention!

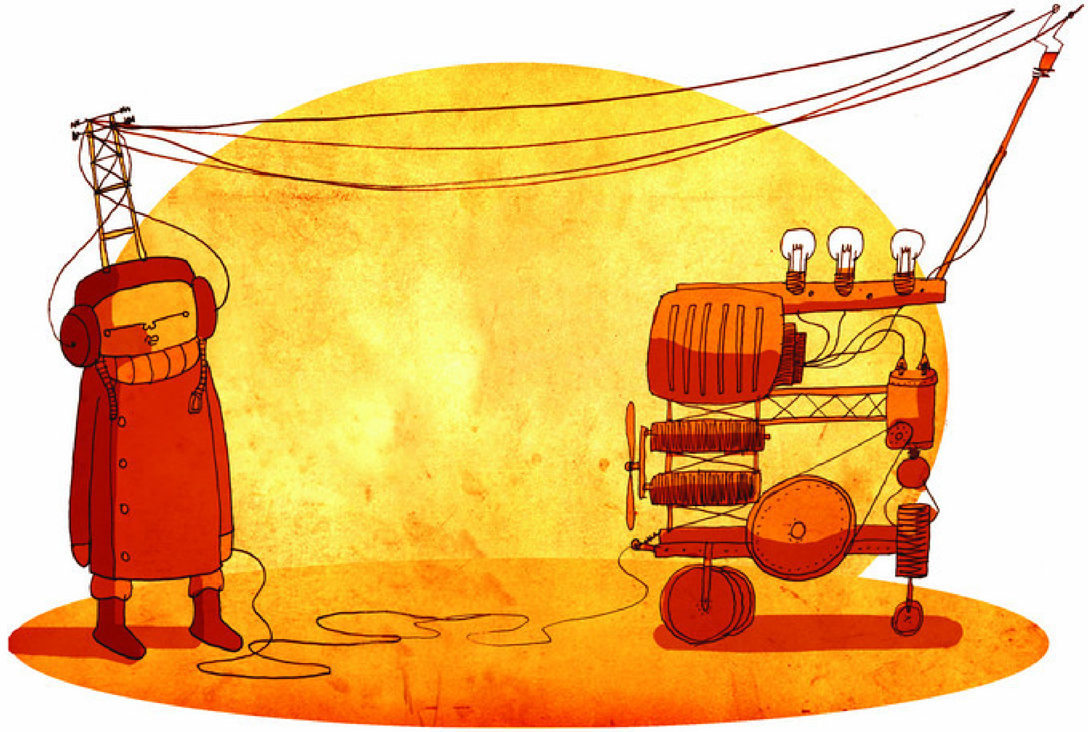


Image by Ben Scruton of the Young Creatives Network. The future's bright.