New algorithms for the strength of graphs

Jérôme Galtier, Orange Labs

LIRMM

June 11, 2009

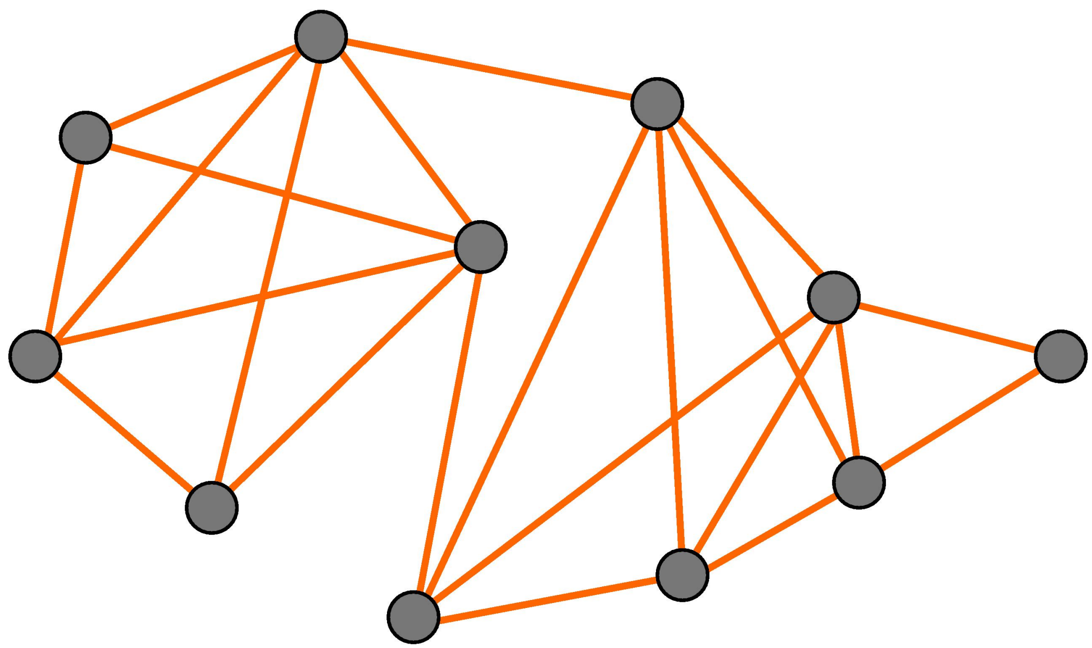
what is the strength of a graph?

Given a graph $G = (V, E)$, let $\mathcal{P}(V)$ be the set of partitions of V, and compute

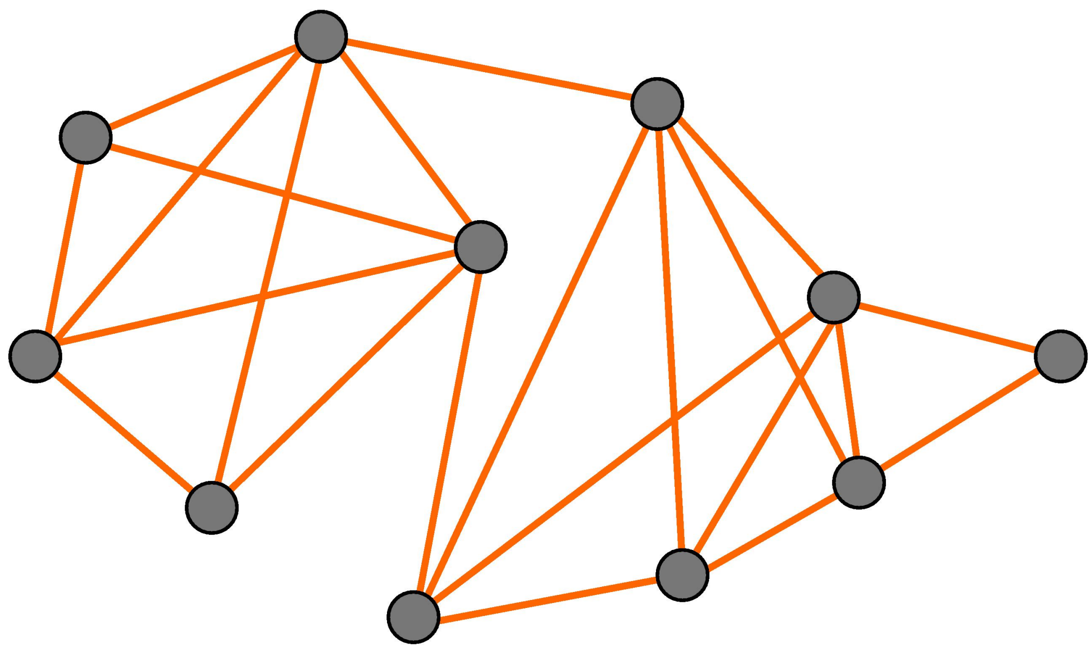
$$\sigma(G) = \min_{\Pi \in \mathcal{P}(V)} \frac{\partial \Pi}{|\Pi| - 1},$$

where $\partial \Pi \subseteq E$ represents the edges between sets of Π.

Strength of graphs: intuition

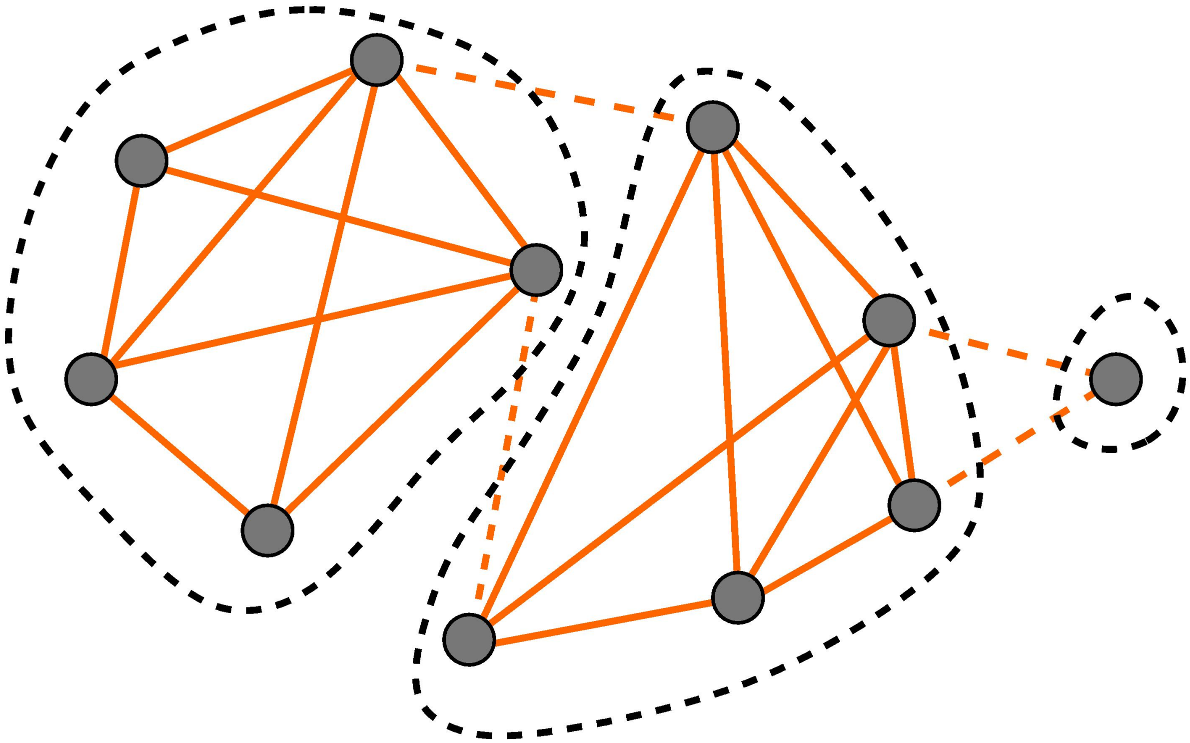


Strength of graphs: intuition

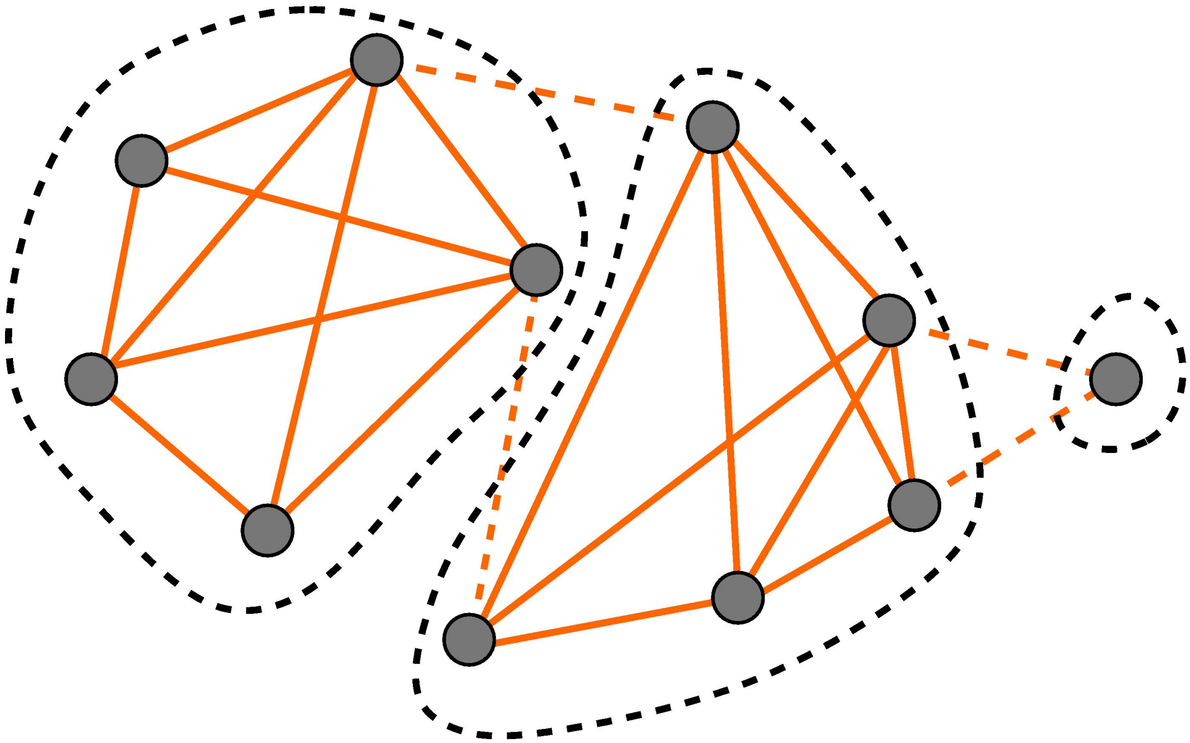


$\longrightarrow$ minimize the ratio $\dfrac{\text{edges withdrawn}}{\text{created components}}$.

Strength of graphs: intuition



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 (Karger 2000).

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	$O(n^3m)$	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	$O(m^2\log(n)^2/\varepsilon^2)$	Within $1+\varepsilon$
2008	G.	$O(n\sigma(G)\log(n)^2/\varepsilon^2)$	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 $w(e) = \delta = O(n^{-3/\varepsilon})$,

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 $w(e) = \delta = O(n^{-3/\varepsilon})$,

(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 $w(e) = \delta = O(n^{-3/\varepsilon})$,

(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 $w(e) = \delta = O(n^{-3/\varepsilon})$,

(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 $w(e) = \delta = O(n^{-3/\varepsilon})$,

(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 $w(e) = \delta = O(n^{-3/\varepsilon})$,

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

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

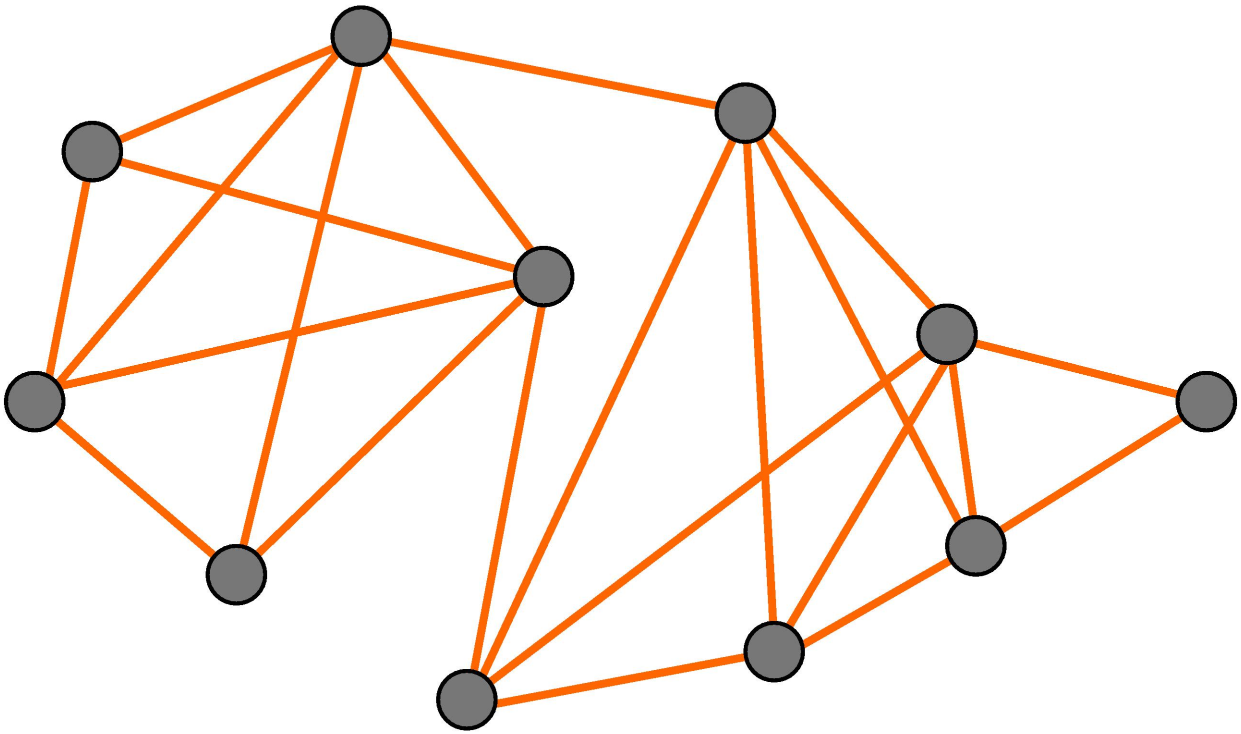
(3) If $w(T) < 1$ go to (1),

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

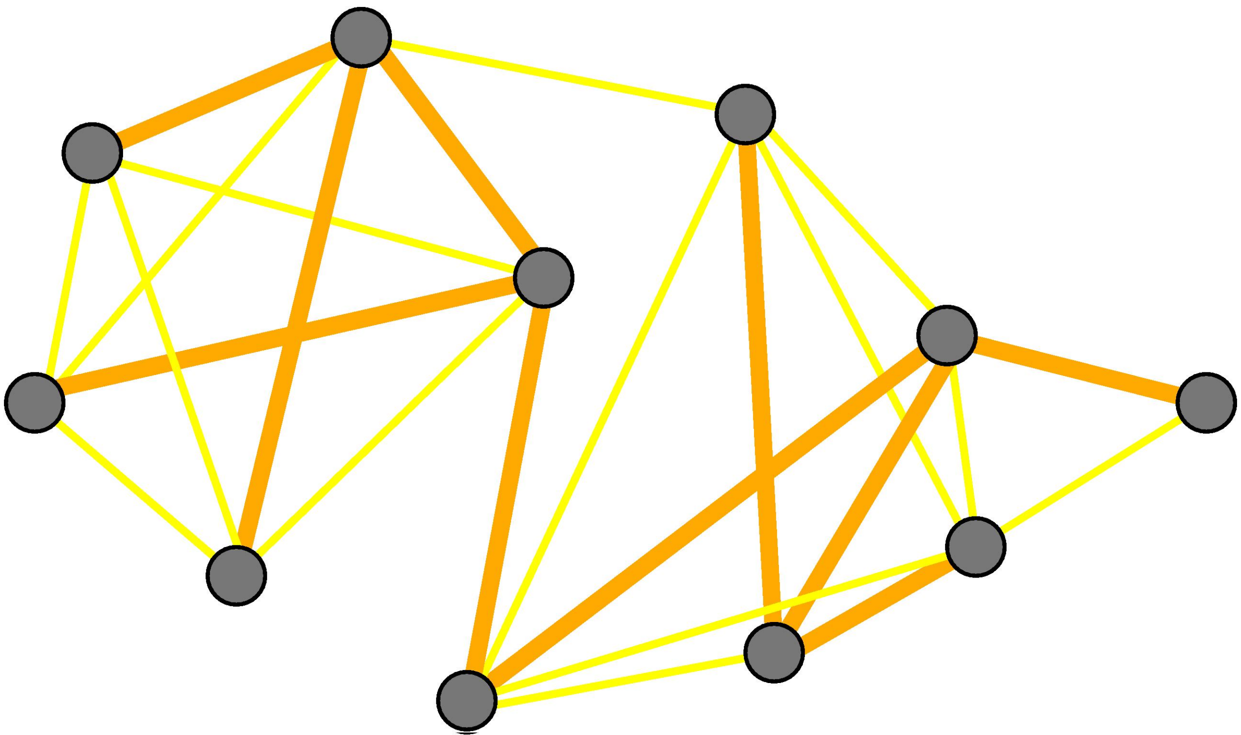
$\rightarrow$ this is an-$(1 + \varepsilon)$ approximation

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

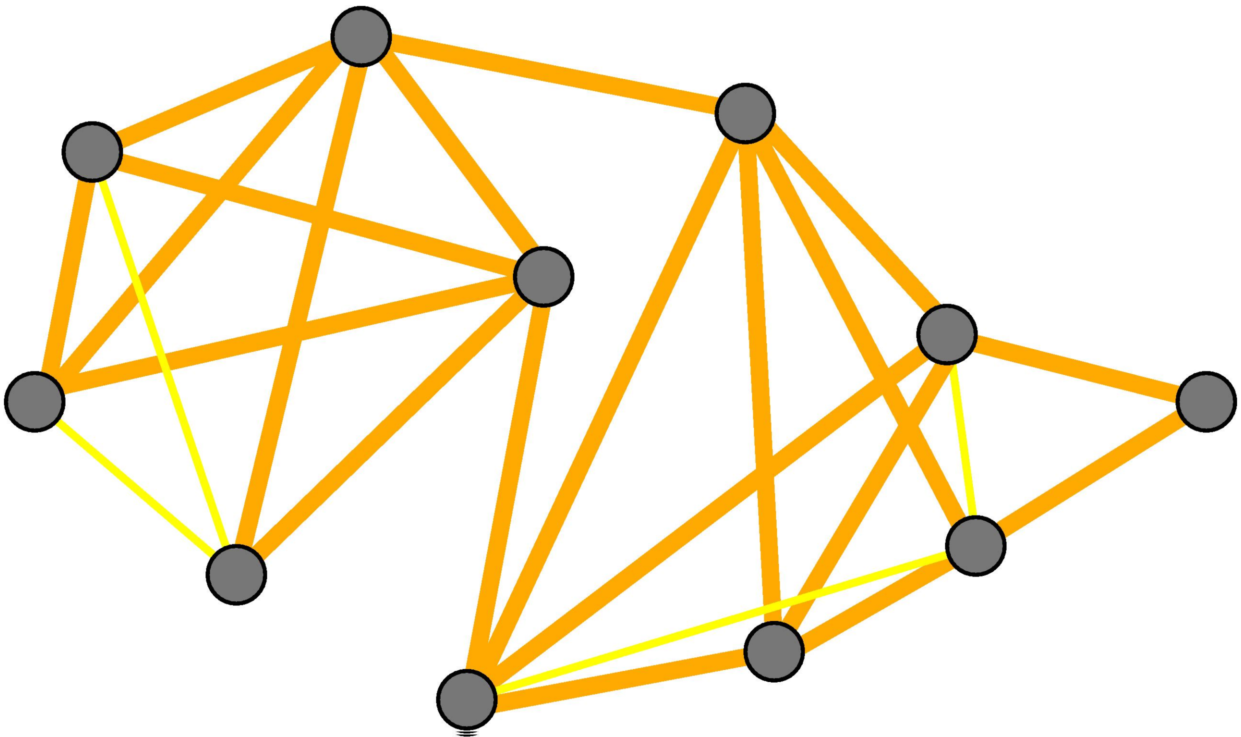
animation of the algorihtm



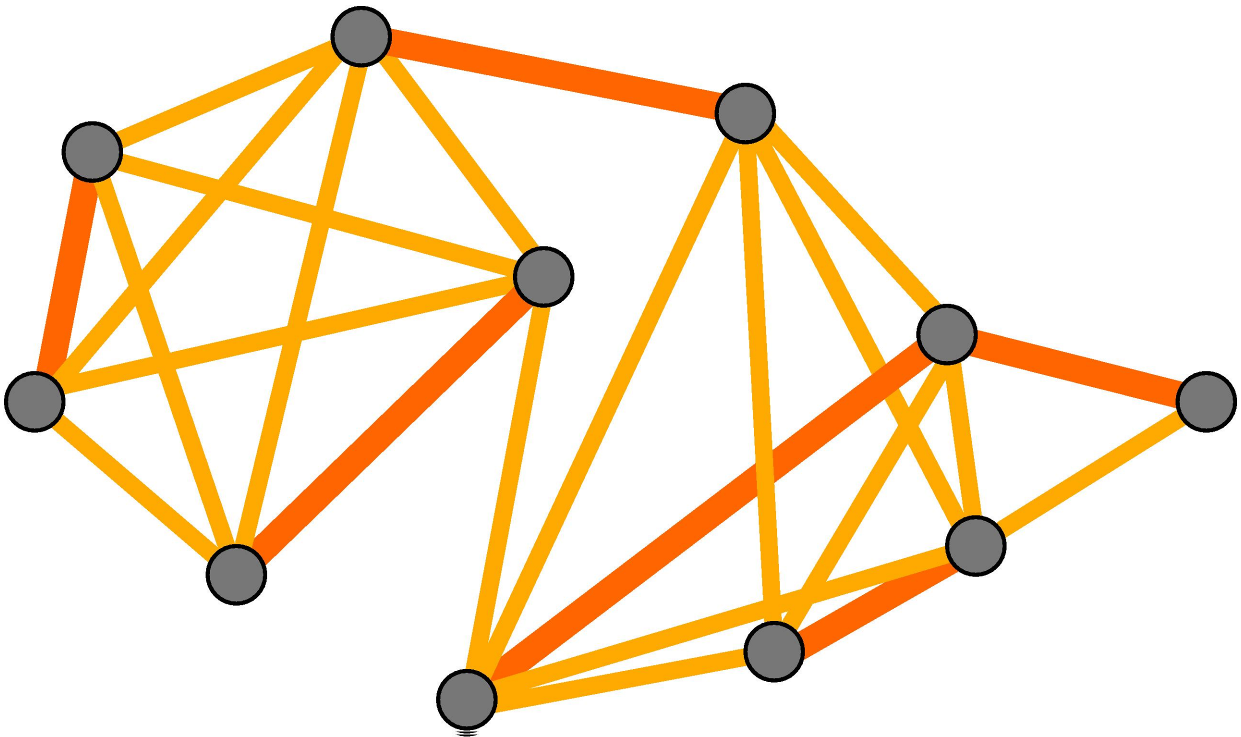
animation of the algorihtm



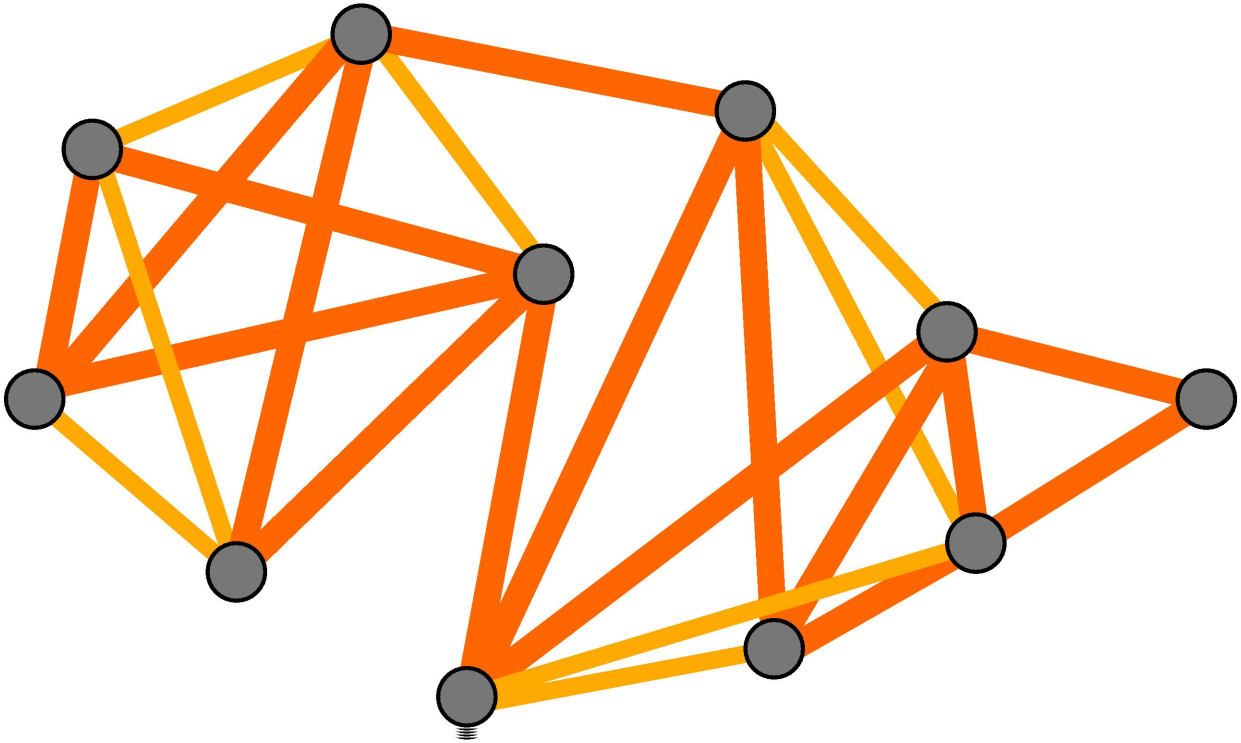
animation of the algorihtm



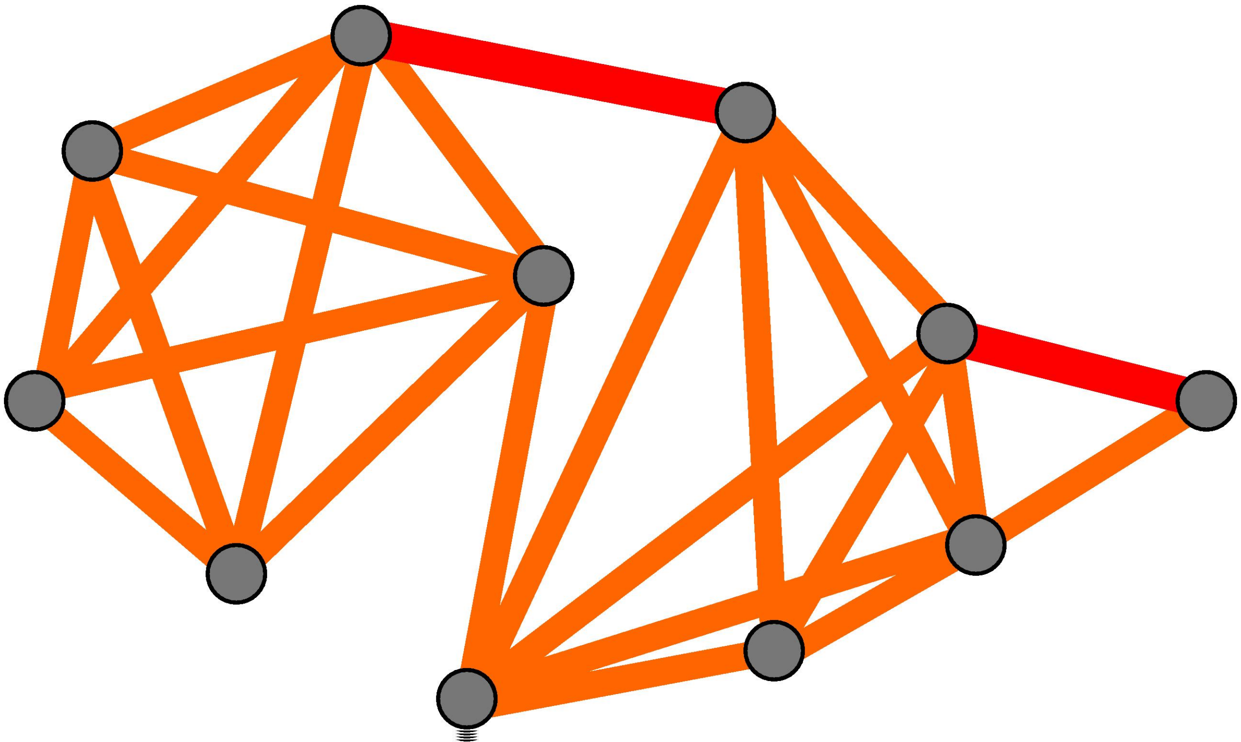
animation of the algorihtm



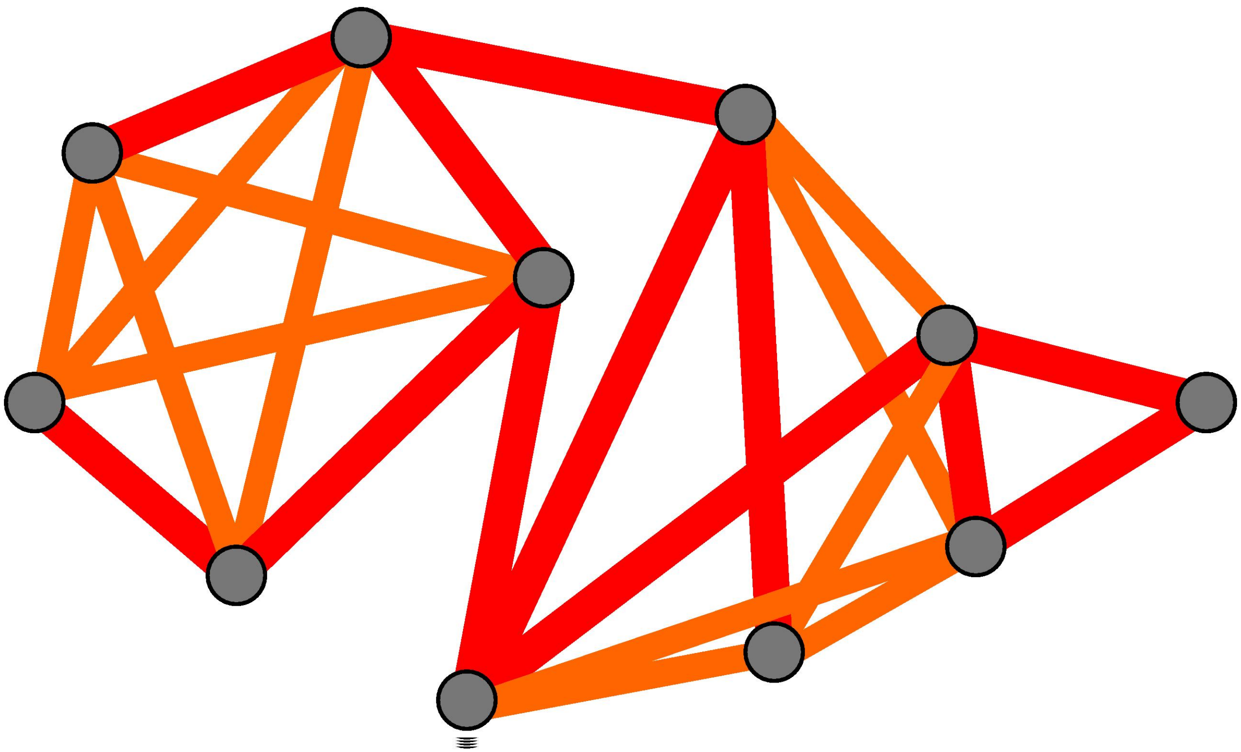
animation of the algorihtm



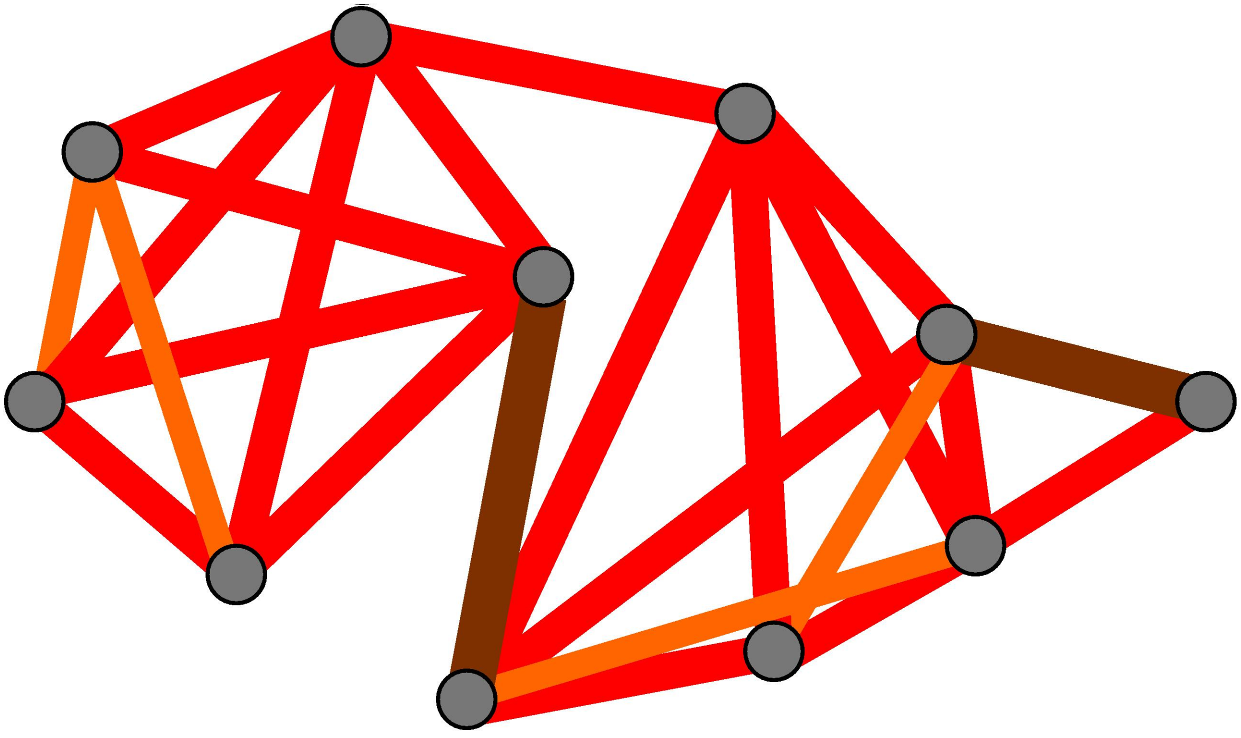
animation of the algorihtm



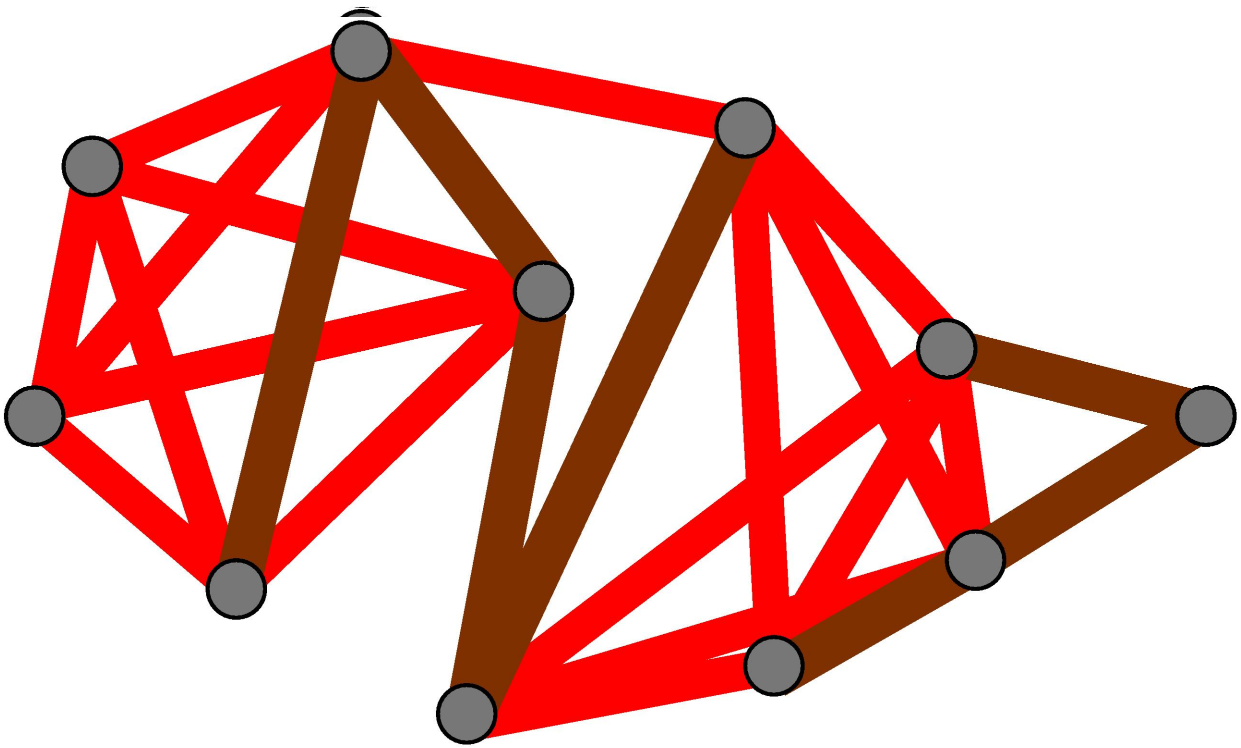
animation of the algorihtm



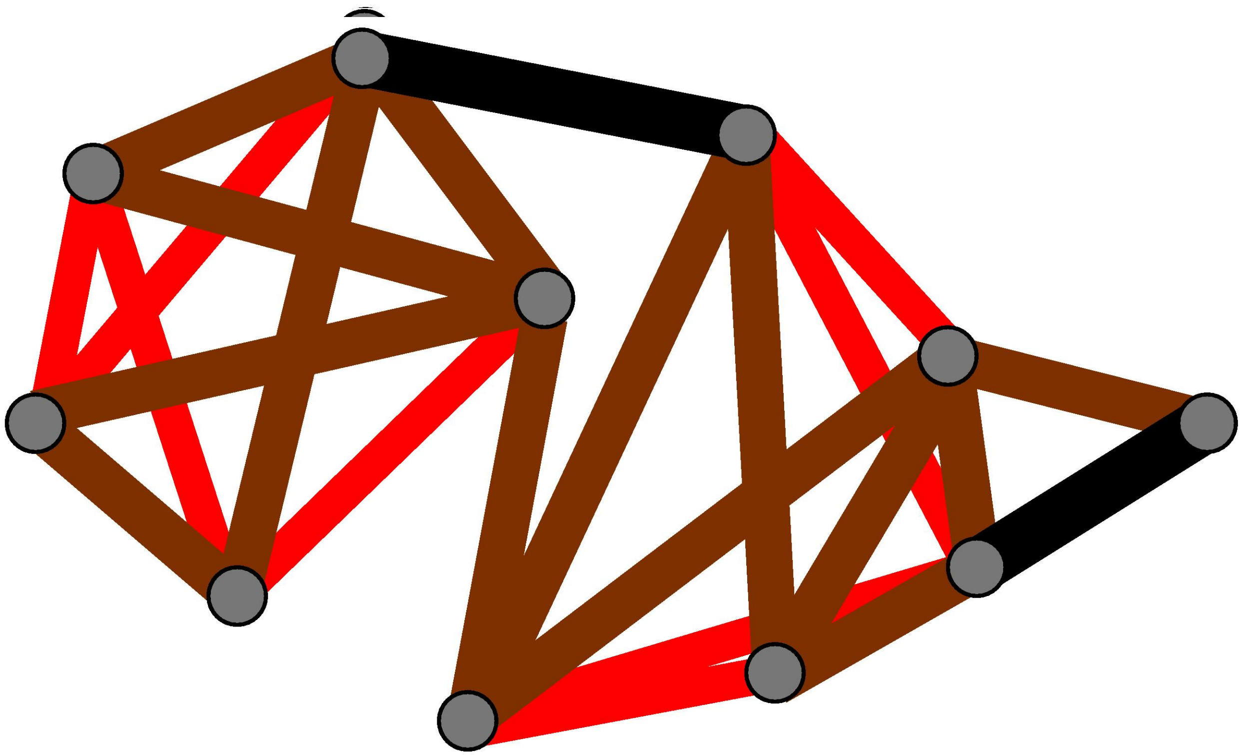
animation of the algorihtm



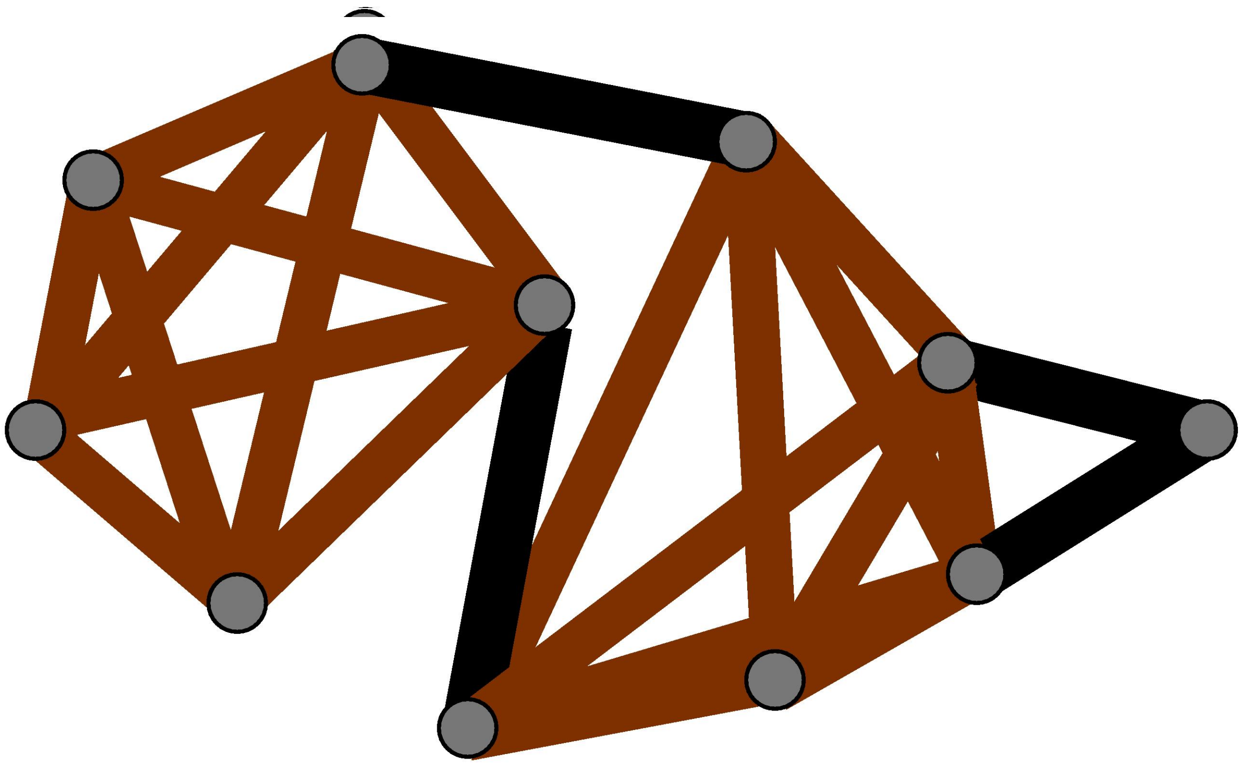
animation of the algorihtm



animation of the algorihtm



animation of the algorihtm



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 $\frac{\log(\delta)}{\log(1+\varepsilon)} = O(\frac{\log(n)}{\varepsilon^2})$,

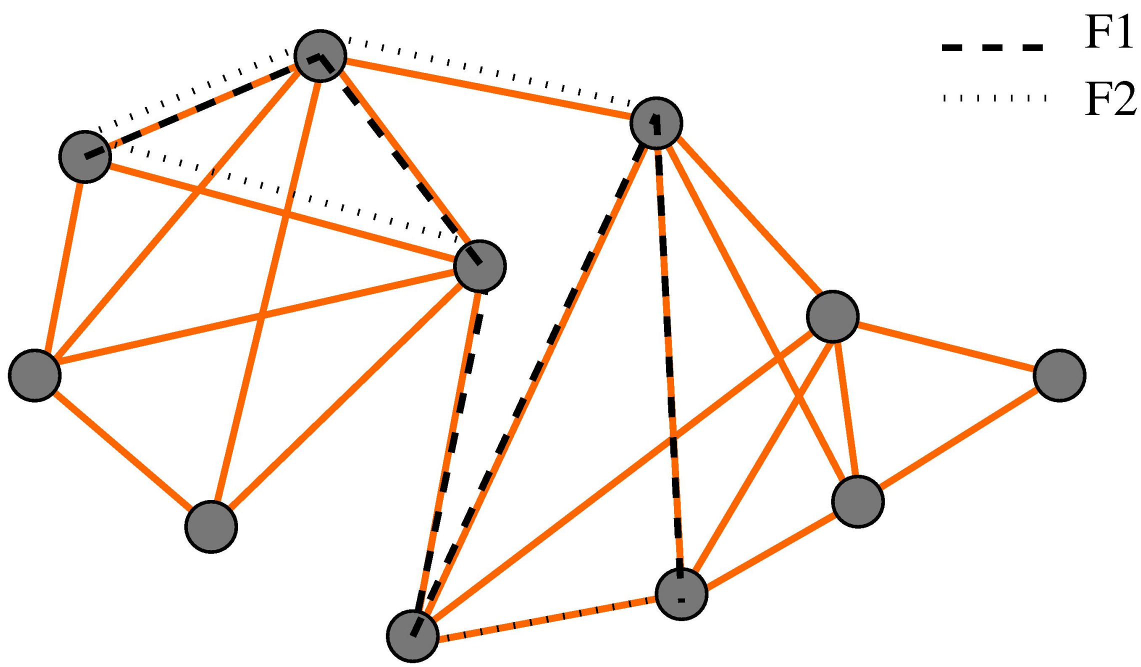
Each step updates $n-1$ edges and runs in $O(m \log(n))$,

$\longrightarrow$ 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.

$\longrightarrow$ 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 $e \in E$.

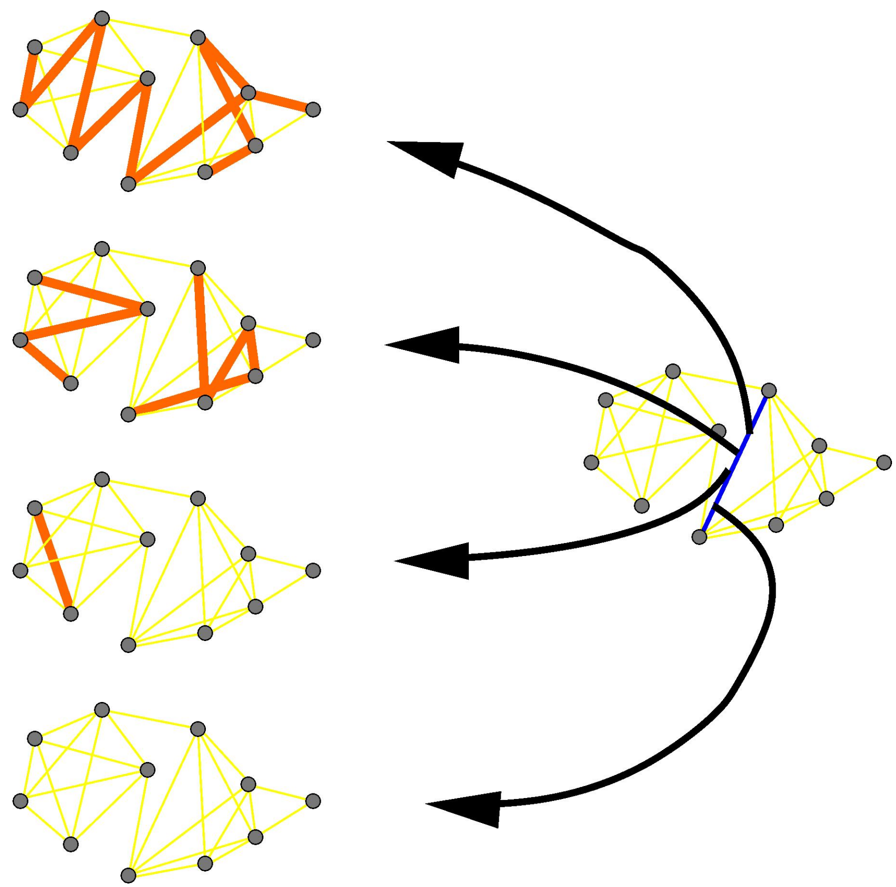
augment the first F_i such that e is independent to F_i.

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

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

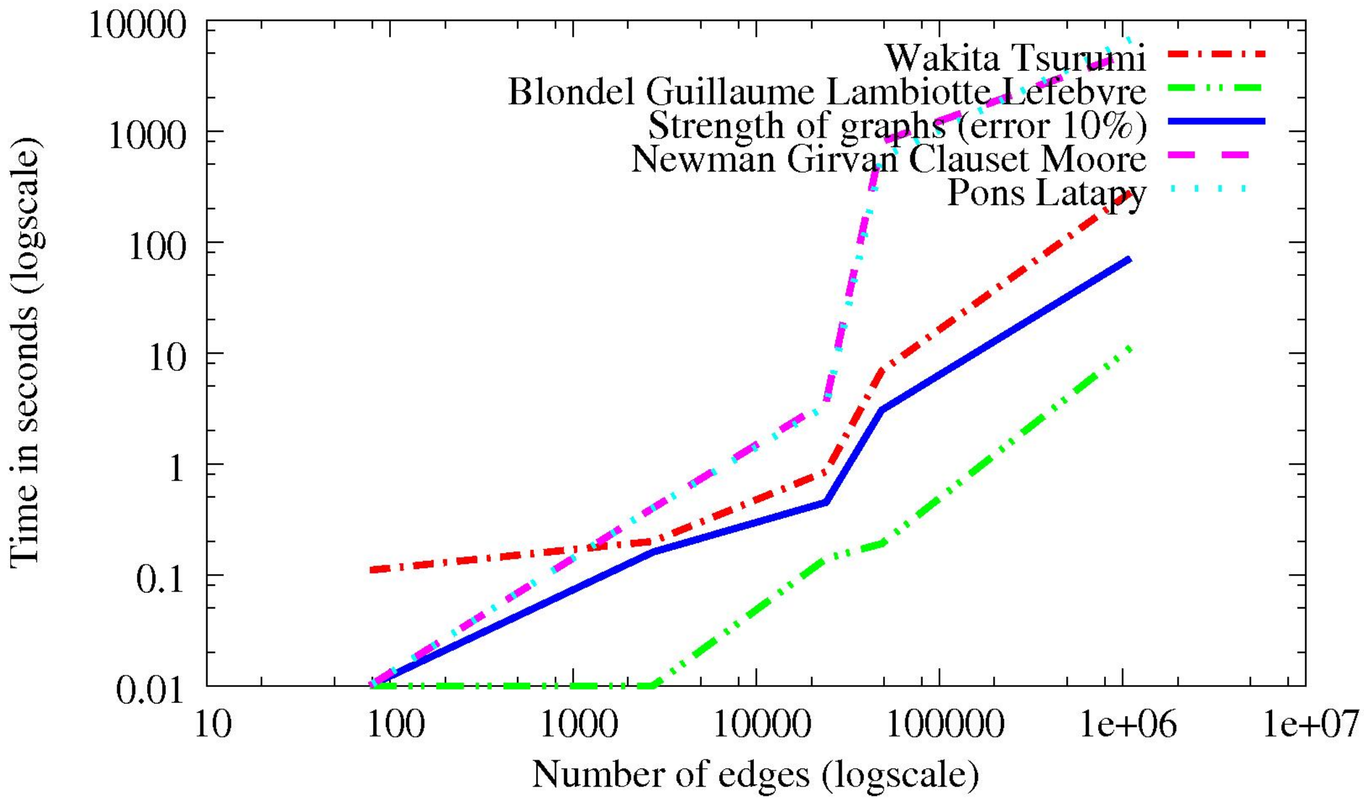
this gives an $O(m \log(n)^3/\varepsilon^2)$ algorithm.

princip illustration

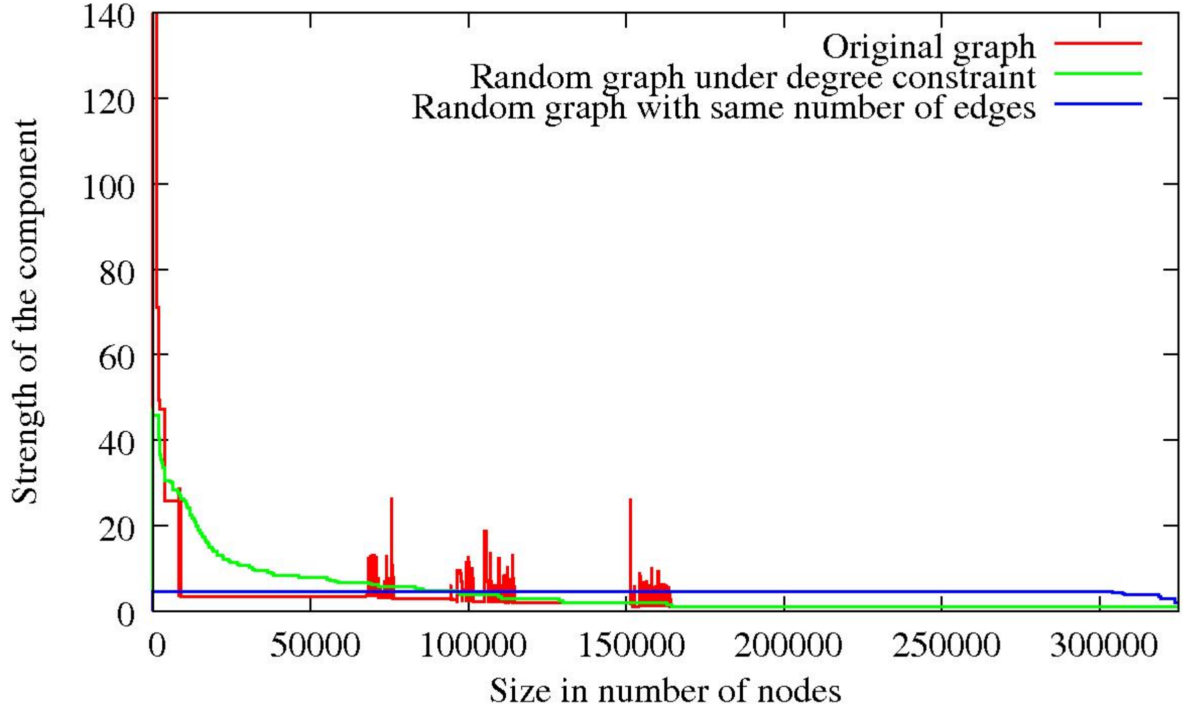


computational linearity of the approximation

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



Spectrum of the web



Conlusions

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 (less than 30 minutes of computation for 10% precision).

Conlusions

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 (less than 30 minutes of computation for 10% precision).

Questions

- can we reduce the ε^2 factor in practice?

Conlusions

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 (less than 30 minutes of computation for 10% precision).

Questions

- can we reduce the ε^2 factor in practice?
- can the polyhedral formulation be further simplified?

Conlusions

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 (less than 30 minutes of computation for 10% precision).

Questions

- can we reduce the ε^2 factor in practice?
- can the polyhedral formulation be further simplified?
- can we win a battle against the logarithms?

Thanks for four attention!!!

(and may the strength be with us)