cadmo
center for algorithms, discrete mathematics & optimization

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Geometric Inhomogeneous Random Graphs (GIRGs)

Johannes Lengler (ETH Zürich)

joint work with K. Bringmann, R. Keusch, C. Koch

Motivation: Network Models

- want to develop good algorithms for large *real-world networks*
  want to have asymptotic statements, benchmarks, …

- real network data is *scarce* and *hard to obtain*
  social: facebook, twitter, mobile phone, friendship, collaboration..
  technological: internet, www, web of things,…

- these networks *share many properties*
  power law degrees, (ultra-)small world, strong clustering, small separators,…

Motivation: Network Models

properties / models	power law degree?	small world?	cluster-ing?	non-rigid clust.?	easy to analyze?
Kleinberg	✗	✓	(✓)	(✓)	✓
pref. attachm.	✓	✓	✗	✗	✗
Chung-Lu	✓	✓	✗	✗	✓
geom. random	✗	✗	✓	✗	✓
hyperbolic	✓	✓	✓	✗	✗
spatial pref. att.	✓	✓	✓	✗	✗
GIRGs	✓	✓	✓	✓	✓

Motivation: Hyperbolic Random Graphs

- best model so far: *hyperbolic random graphs*
- each vertex draws a random position in a hyperbolic disc of radius R.
- two points connect if and only if their distance is at most R.
- has many nice properties:
  power law degrees, clustering, small world, …

Motivation: Hyperbolic Random Graphs

- best model so far: *hyperbolic random graphs*
- each vertex draws a random position in a hyperbolic disc of radius R.
- two points connect if and only if their distance is at most R.
- has many nice properties:
  power law degrees, clustering, small world, …

BUT: kind of complicated…. 😧

$$\bar{k}(r) = \frac{N}{2\pi(\cosh R - 1)} \left\{ 2\pi(\cosh R - 1) - 2\cosh R \left(\arcsin \frac{\tanh(r/2)}{\tanh R} + \arctan \frac{\cosh R \sinh(r/2)}{\sqrt{\sinh(R + r/2)\sinh(R - r/2)}} \right) \right.$$
$$\left. + \arctan \frac{(\cosh R + \cosh r)\sqrt{\cosh 2R - \cosh r}}{\sqrt{2}(\sinh^2 R - \cosh R - \cosh r)\sinh(r/2)} - \arctan \frac{(\cosh R - \cosh r)\sqrt{\cosh 2R - \cosh r}}{\sqrt{2}(\sinh^2 R + \cosh R - \cosh r)\sinh(r/2)} \right\}, \quad (11)$$

Motivation: GIRGs

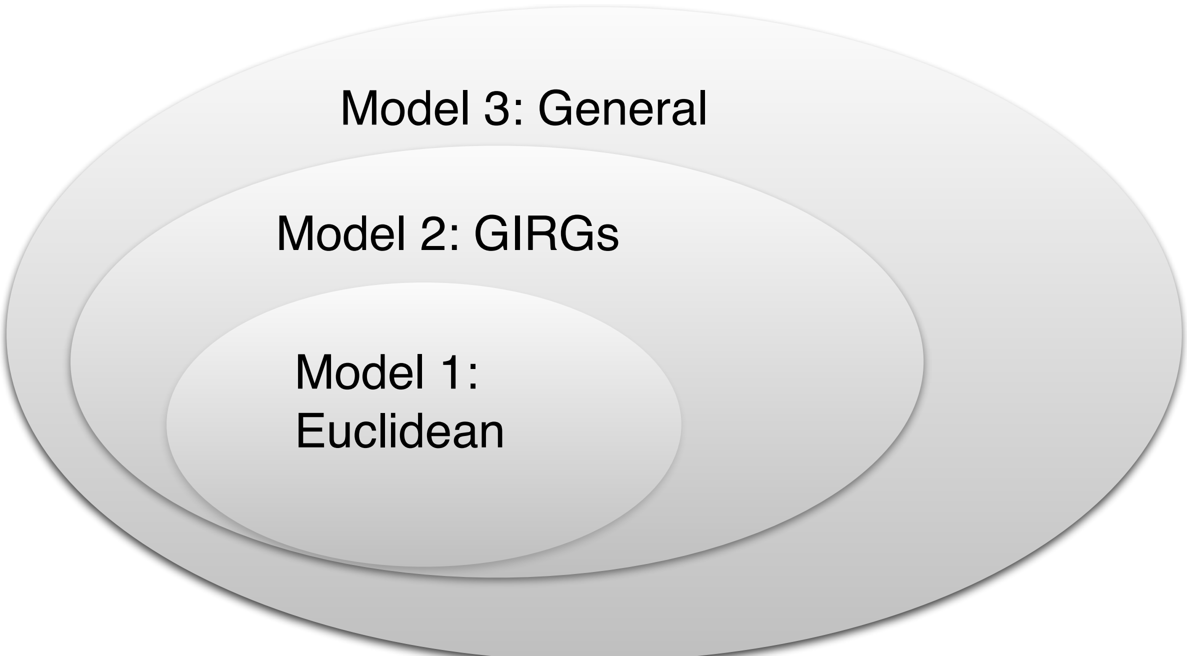
GIRGs (Geometric Inhomogeneous Random Graphs)

- are *natural.*
- *are* very *easy to analyze*.
- are extremely *flexible*.

Motivation: GIRGs

GIRGs (Geometric Inhomogeneous Random Graphs)

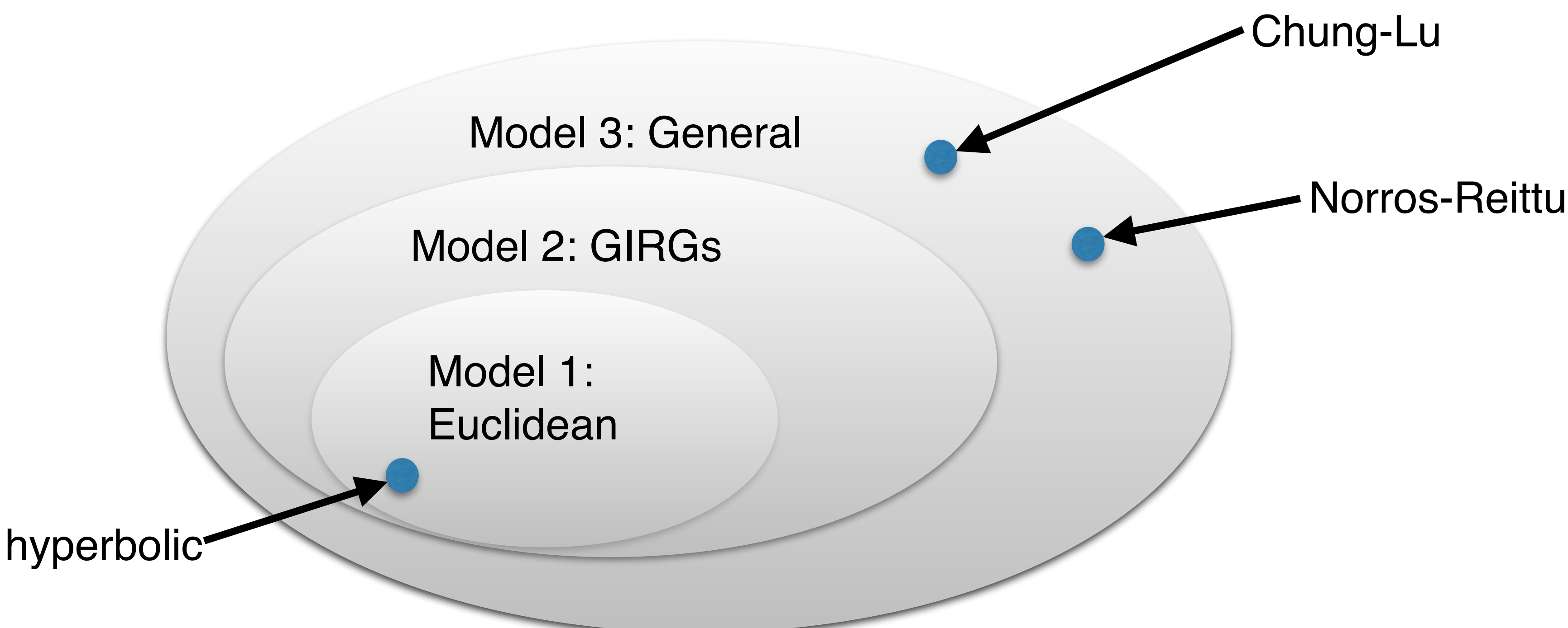
- are *natural.*
- *are* very *easy to analyze*.
- are extremely *flexible*.



Motivation: GIRGs

GIRGs (Geometric Inhomogeneous Random Graphs)

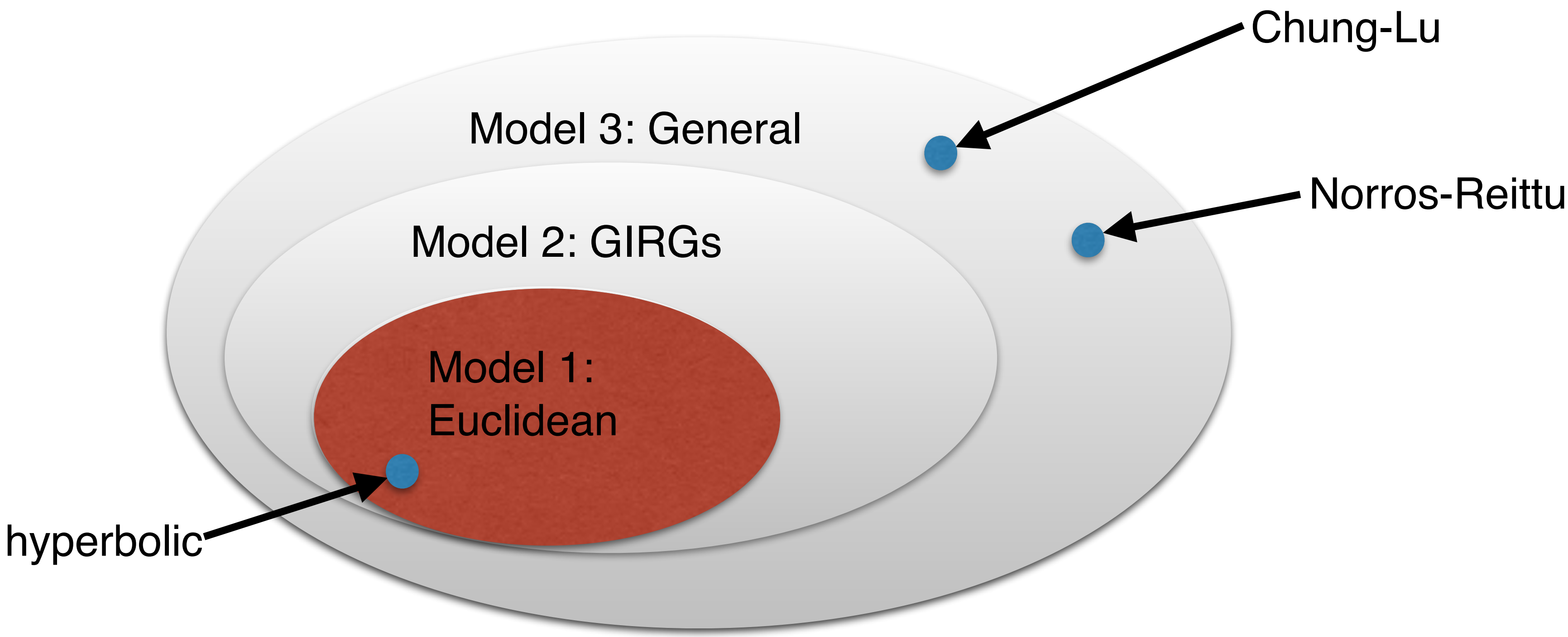
- are *natural.*
- *are* very *easy to analyze*.
- are extremely *flexible*.



Motivation: GIRGs

GIRGs (Geometric Inhomogeneous Random Graphs)

- are *natural.*
- *are* very *easy to analyze*.
- are extremely *flexible*.



Model 1: Euclidean

Model 1: Euclidean

- We start with n vertices.

Model 1: Euclidean

- We start with n vertices.
- Each vertex v_i draws independently a weight w_i from a power law distribution:

$$\Pr[w_i = w] = \Theta(w^{-\beta}), \quad \text{where } 2 < \beta < 3.$$

Model 1: Euclidean

- We start with n vertices.
- Each vertex v_i draws independently a weight w_i from a power law distribution:

$$\Pr[w_i = w] = \Theta(w^{-\beta}), \quad \text{where } 2 < \beta < 3.$$

- Each vertex v_i draws independently a position x_i from the hypercube $[0,1]^d$.

Model 1: Euclidean

- We start with n vertices.

- Each vertex v_i draws independently a weight w_i from a power law distribution:

$$\Pr[w_i = w] = \Theta(w^{-\beta}), \quad \text{where } 2 < \beta < 3.$$

- Each vertex v_i draws independently a position x_i from the hypercube $[0,1]^d$.

- For each pair (i,j), we *independently* connect v_i and v_j with prob.

$$p_{i,j} = p(w_i, w_j, x_i, x_j) = \Theta\left(\min\left\{1, |x_i - x_j|^{-d\alpha}\left(\frac{w_i w_j}{n}\right)^{\alpha}\right\}\right),$$

where $\alpha > 1$ is a parameter.

Basic Properties

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Lemma: For any fixed x_i, w_i, w_j,

1. $\Pr_{x_j}[v_i \sim v_j] = \Theta\left(\min\left\{1, \frac{w_i w_j}{n}\right\}\right)$.
2. $\mathbb{E}[\deg(v_i)] = \Theta(w_i)$.

Basic Properties

Lemma: For any fixed x_i, w_i, w_j,

1. $\Pr_{x_j}[v_i \sim v_j] = \Theta\left(\min\left\{1, \frac{w_i w_j}{n}\right\}\right)$.
2. $\mathbb{E}[\deg(v_i)] = \Theta(w_i)$.

Corollary:

- The degree of a vertex v_i of weight w_i is Poisson distributed (in the limit) with mean $\Theta(w_i)$.
- $\mathbb{E}[w_i] = \Theta(1)$ => There are O(n) edges.

(Ultra-)Small World

Theorem: Whp,

1. the graph contains a giant component of linear size.
2. all other components are of polylog size.
3. the diameter of the graph is polylogarithmic.
4. the average distance in the giant is $(2+o(1))\frac{\log\log n}{|\log(\beta-2)|}$.

(Ultra-)Small World

Theorem: Whp,

1. the graph contains a giant component of linear size.
2. all other components are of polylog size.
3. the diameter of the graph is polylogarithmic.
4. the average distance in the giant is $(2+o(1))\frac{\log\log n}{|\log(\beta-2)|}$.

- holds in the most general model (including Chung-Lu graphs)
- same is true for other power-law graph models (e.g., preferential attachment)

Clustering

Definition:
The *clustering coefficient* of a graph is

$$\mathrm{cc} := \Pr_{u,v,w}[v \sim w \mid u \in V, v, w \in \Gamma(u)].$$

Clustering

Definition:
The *clustering coefficient* of a graph is

$$\mathrm{cc} := \Pr_{u,v,w}[v \sim w \mid u \in V, v, w \in \Gamma(u)].$$

- Social (and other) networks have large clustering coefficient.
- most models with power law degrees have $\mathrm{cc} = \Theta(1/n)$.
  (Chung-Lu, preferential attachment, …)
- *exception*: hyperbolic random graphs have $\mathrm{cc} = \Omega(1)$.

Clustering

Definition:
The *clustering coefficient* of a graph is

$$\mathrm{cc} := \Pr_{u,v,w}[v \sim w \mid u \in V, v, w \in \Gamma(u)].$$

- Social (and other) networks have large clustering coefficient.
- most models with power law degrees have $\mathrm{cc} = \Theta(1/n)$. (Chung-Lu, preferential attachment, …)
- *exception*: hyperbolic random graphs have $\mathrm{cc} = \Omega(1)$.

Theorem: GIRGs have $\mathrm{cc} = \Omega(1)$.

Stability

Theorem: GIRGs have *small separators:*

It suffices to delete $n^{1-\Omega(1)}$ edges from the graph to split the giant into two components of linear size.

Stability

Theorem: GIRGs have *small separators:*

It suffices to delete $n^{1-\Omega(1)}$ edges from the graph to split the giant into two components of linear size.

- was unstudied for hyperbolic random graphs.
- Chung Lu and pref. attachment models are different: Removing $o(n)$ edges or vertices reduces the giant by at most $o(n)$.
- Real-world networks have small separators.

Entropy/Compression

Observation:
The web graph can be stored using bits per edge.

Entropy/Compression

Observation:
The web graph can be stored using 2-3 (!) bits per edge.

Entropy/Compression

Observation:
The web graph can be stored using 2-3 (!) bits per edge.

Theorem: We can store a GIRG with expected $O(n)$ bits, so that we can answer the queries
- "What is the degree of v?"
- "What is the i-th neighbor of v?"

in time $O(1)$. The algorithm has expected runtime $O(n)$.

Entropy/Compression

Observation:
The web graph can be stored using 2-3 (!) bits per edge.

Theorem: We can store a GIRG with expected $O(n)$ bits, so that we can answer the queries
- "What is the degree of v?"
- "What is the i-th neighbor of v?"

in time $O(1)$. The algorithm has expected runtime $O(n)$.

- compression algorithm for hyperbolic graphs was known.
- Chung Lu and pref. attachment models have entropy $\Theta(n \log n)$. (I.e., need $\Theta(\log n)$ bits per edge.)

Sampling

Theorem: For every concrete function

$$p(w_i, w_j, x_i, x_j) = \Theta\left(\min\left\{1, \left(|x_i - x_j|^{-d} \cdot \frac{w_i w_j}{n}\right)^{\alpha}\right\}\right),$$

we can sample a GIRG in expected linear time (under some technical assumptions).

Sampling

Theorem: For every concrete function

$$p(w_i, w_j, x_i, x_j) = \Theta\left(\min\left\{1, \left(|x_i - x_j|^{-d} \cdot \frac{w_i w_j}{n}\right)^\alpha\right\}\right),$$

we can sample a GIRG in expected linear time (under some technical assumptions).

- Naive sampling needs time $\Theta(n^2)$.
- Efficient algorithms were known for Chung-Lu model and others.
- Best previous algorithm for hyperbolic random graphs had runtime $\Theta(n^{3/2})$.

Greedy Routing

Greedy Routing

- Vertex s wants to send message to vertex t.
- s only knows position and weight of its neighbors and of t

Greedy Routing

- Vertex s wants to send message to vertex t.
- s only knows position and weight of its neighbors and of t
- We try to maximize greedily $\varphi(v) := \Pr[t \text{ is neighbor of } v]$.

Greedy Routing

- Vertex s wants to send message to vertex t.
- s only knows position and weight of its neighbors and of t

- We try to maximize greedily $\varphi(v) := \Pr[t \text{ is neighbor of } v]$.

- ALGORITHM (greedy routing):
  REPEAT until we find t:
  - s' := best neighbor of s
  - IF $\varphi(s') > \varphi(s)$ THEN $s' := s$ ELSE *fail.*

Greedy Routing

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

- Vertex s wants to send message to vertex t.
- s only knows position and weight of its neighbors and of t
- We try to maximize greedily $\varphi(v) := \Pr[t \text{ is neighbor of } v]$.
- ALGORITHM (greedy routing):
  REPEAT until we find t:
  - s' := best neighbor of s
  - IF $\varphi(s') > \varphi(s)$ THEN $s' := s$ ELSE *fail.*

Theorem: With probability $\Omega(1)$, greedy routing succeeds

in $(2+o(1))\dfrac{\log\log n}{|\log(\beta-2)|}$ steps.

With small modifications (e.g. backtracking), it succeeds within this time whp and in expectation.

Bootstrap Percolation

- We fix a region B of volume .
- In round 0, every vertex in B turns active with probability p.
- An active vertex stays active forever.
- A vertex has with k active neighbors turns active next round.

Bootstrap Percolation

- We fix a region B of volume .
- In round 0, every vertex in B turns active with probability p.
- An active vertex stays active forever.
- A vertex has with k active neighbors turns active next round.

Theorem: Let $p_0 := \nu^{-1/(\beta-1)}$. Then
- if $p \gg p_0$ then $\Theta(n)$ vertices turn active whp;
- if $p \ll p_0$ then no vertex turns active after round 0 whp;
- if $p = \Theta(p_0)$ then either case happens with prob $\Omega(1)$.

Bootstrap Percolation

- We fix a region B of volume .
- In round 0, every vertex in B turns active with probability p.
- An active vertex stays active forever.
- A vertex has with k active neighbors turns active next round.

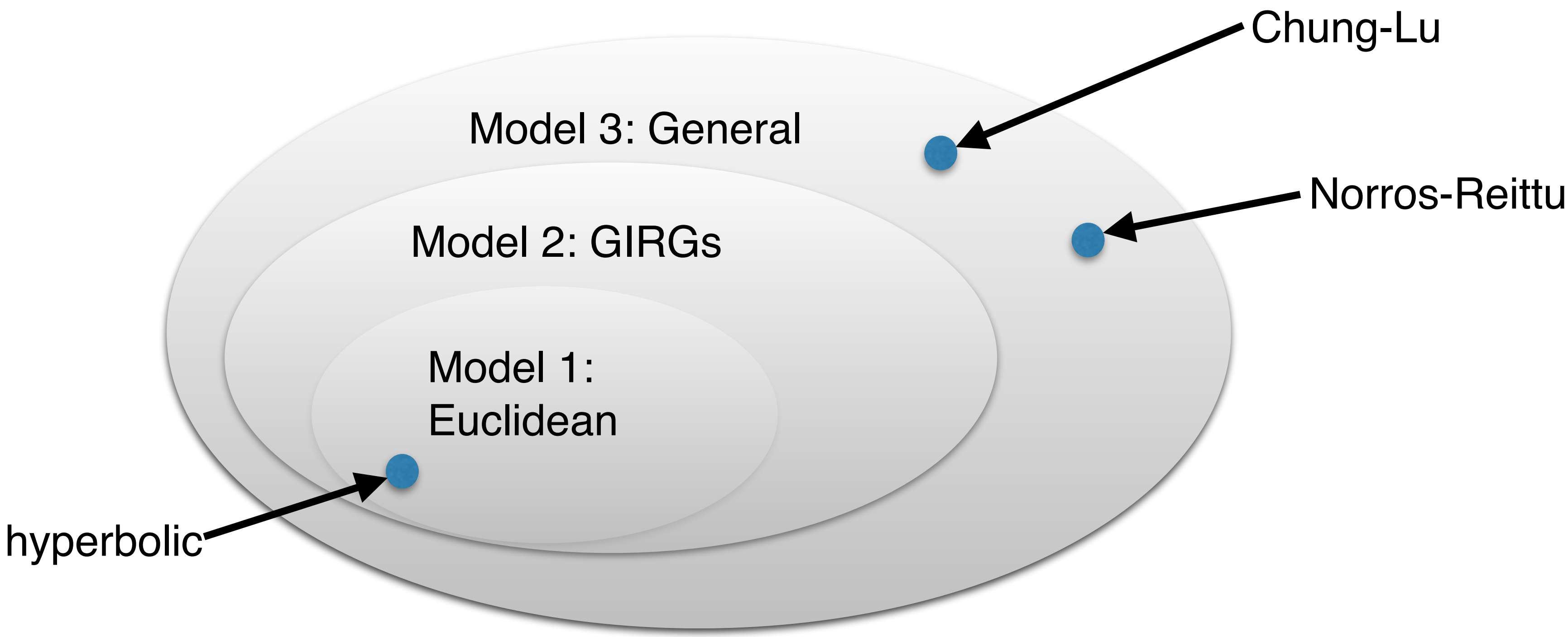
Theorem: Let $p_0 := \nu^{-1/(\beta-1)}$. Then

- if $p \gg p_0$ then $\Theta(n)$ vertices turn active whp;
- if $p \ll p_0$ then no vertex turns active after round 0 whp;
- if $p = \Theta(p_0)$ then either case happens with prob $\Omega(1)$.

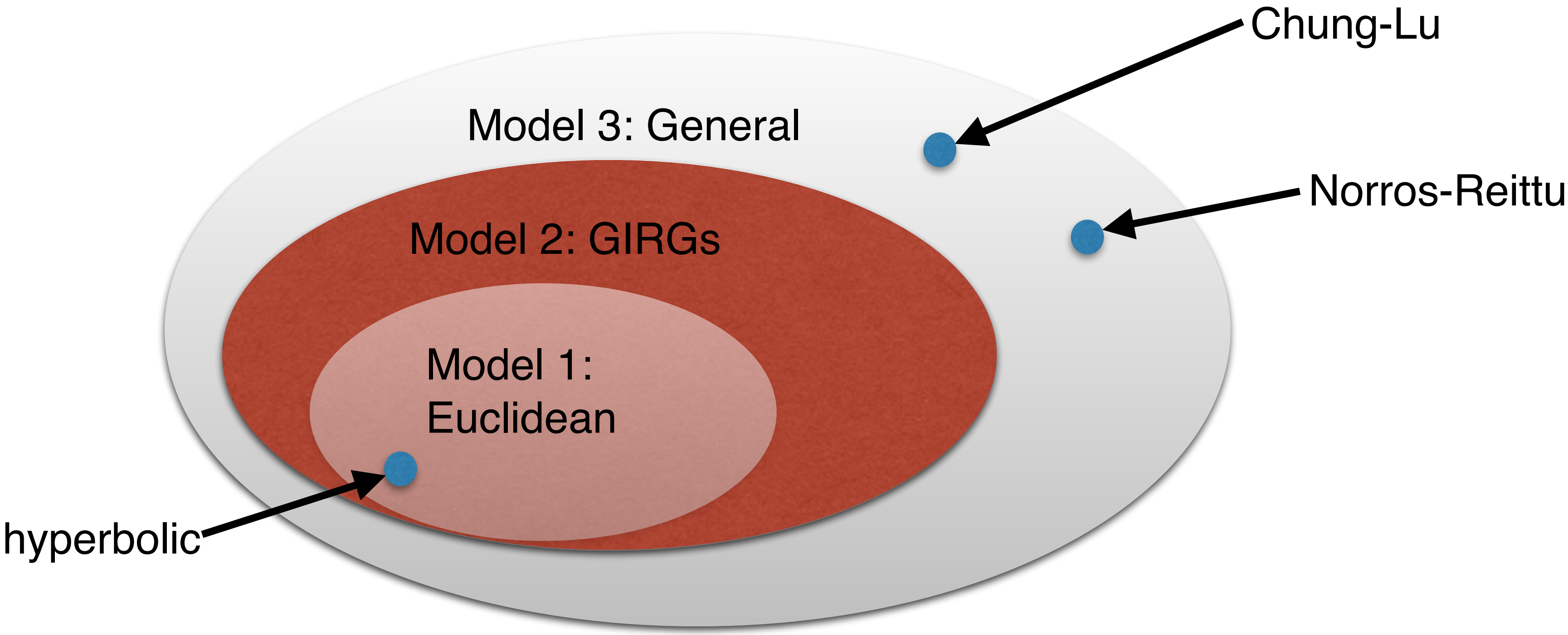
Theorem: Assume $\alpha > \beta - 1$. Let v be a vertex of weight $w \gg 1$ and distance $r \geq \dots$ from B. The whp v turns active in round $(1 \pm o(1))\ell(v) \pm O(1)$, where

$$\ell(v) := \begin{cases} \max\{0,\ \log\log_\nu (\|r^d n/w) /|\log(\beta-2)|\}, & \text{if } w > (r^d n)^{1/(\beta-1)}, \\ (2\log\log_\nu(r^d n) - \log\log_\nu w)/|\log(\beta-2)|, & \text{if } w \leq (r^d n)^{1/(\beta-1)}. \end{cases}$$

Non-Euclidean GIRGs



Non-Euclidean GIRGs



Model 1: Euclidean

- We start with n vertices.
- Each vertex v_i draws independently a weight w_i from a power law distribution:

$$\Pr[w_i = w] = \Theta(w^{-\beta}), \quad \text{where } 2 < \beta < 3.$$

- Each vertex v_i draws independently a position x_i from the hypercube $[0,1]^d$.
- For each pair (i,j), we *independently* connect v_i and v_j with prob.

$$p_{i,j} = p(w_i, w_j, x_i, x_j) = \Theta\left(\min\left\{1, \left(|x_i - x_j|^{-d} \cdot \frac{w_i w_j}{n}\right)^{\alpha}\right\}\right),$$

where $\alpha > 1$ is a parameter,

Model 2: Distance

- We start with n vertices.
- Each vertex v_i draws independently a weight w_i from a power law distribution:

$$\Pr[w_i = w] = \Theta(w^{-\beta}), \quad \text{where } 2 < \beta < 3.$$

- Each vertex v_i draws independently a position x_i from the hypercube $[0,1]^d$.
- For each pair (i,j), we *independently* connect v_i and v_j with prob.

$$p_{i,j} = p(w_i, w_j, x_i, x_j) = \Theta\left(\min\left\{1, \left(\mathrm{Vol}(B_{i,j})^{-1} \cdot \frac{w_i w_j}{n}\right)^{\alpha}\right\}\right),$$

where $\alpha > 1$ is a parameter,
and $B_{i,j}$ is the ball around x_i with radius $d(x_i, x_j)$.

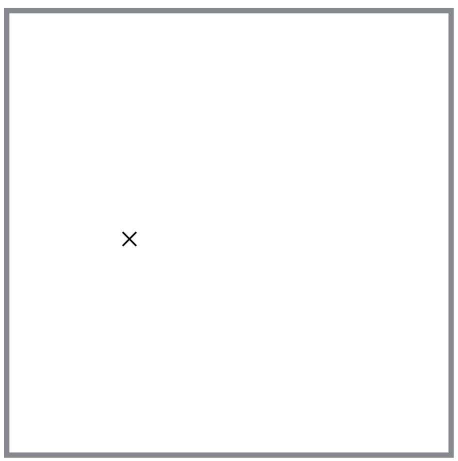
Model 2: Distance

- For each pair (i,j), we *independently* connect v_i and v_j with prob.

$$p_{i,j} = p(w_i, w_j, x_i, x_j) = \Theta\left(\min\left\{1, \left(\mathrm{Vol}(B_{i,j})^{-1} \cdot \frac{w_i w_j}{n}\right)^{\alpha}\right\}\right),$$

where $\alpha > 1$ is a parameter,
and $B_{i,j}$ is the ball around x_i with radius $d(x_i, x_j)$.

Example: minimum component distance



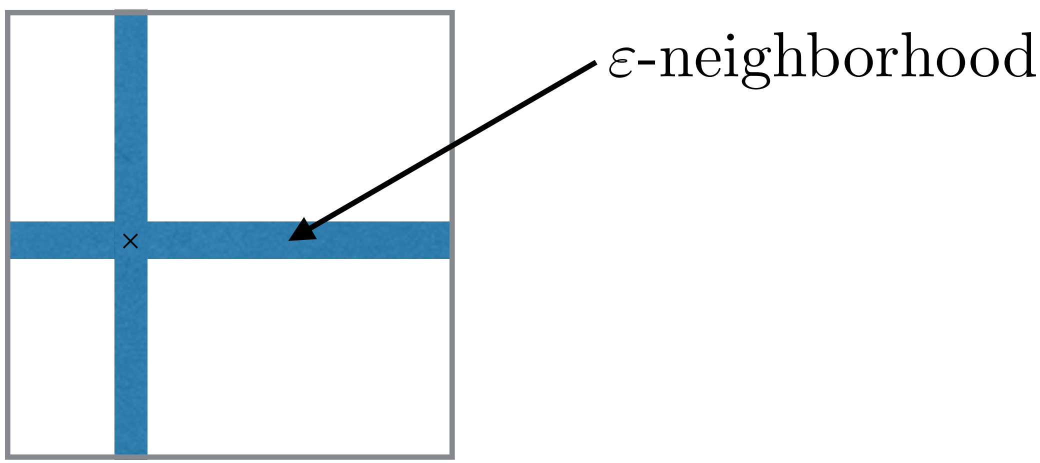
Model 2: Distance

- For each pair (i,j), we *independently* connect v_i and v_j with prob.

$$p_{i,j} = p(w_i, w_j, x_i, x_j) = \Theta\left(\min\left\{1, \left(\mathrm{Vol}(B_{i,j})^{-1} \cdot \frac{w_i w_j}{n}\right)^{\alpha}\right\}\right),$$

where $\alpha > 1$ is a parameter,
and $B_{i,j}$ is the ball around x_i with radius $d(x_i, x_j)$.

Example: minimum component distance



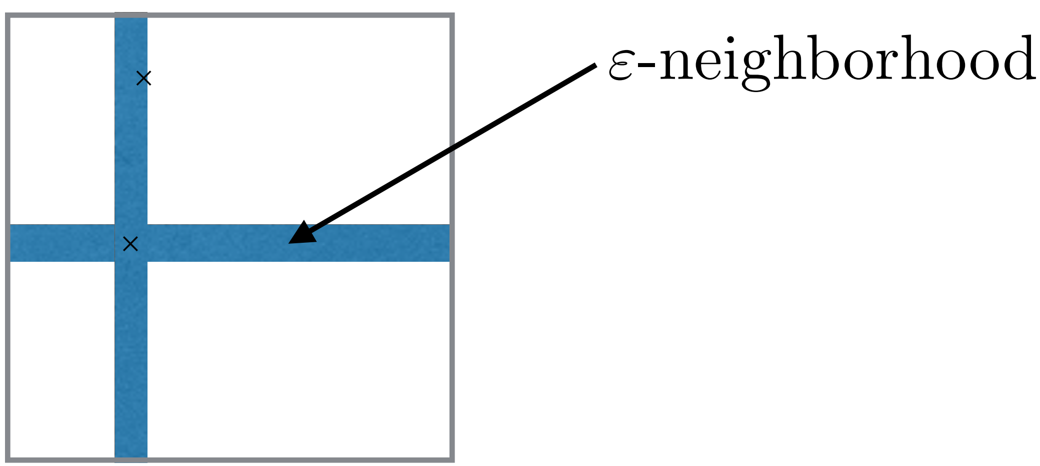
Model 2: Distance

- For each pair (i,j), we *independently* connect v_i and v_j with prob.

$$p_{i,j} = p(w_i, w_j, x_i, x_j) = \Theta\left(\min\left\{1, \left(\mathrm{Vol}(B_{i,j})^{-1} \cdot \frac{w_i w_j}{n}\right)^{\alpha}\right\}\right),$$

  where $\alpha > 1$ is a parameter,
  and $B_{i,j}$ is the ball around x_i with radius $d(x_i, x_j)$.

Example: minimum component distance



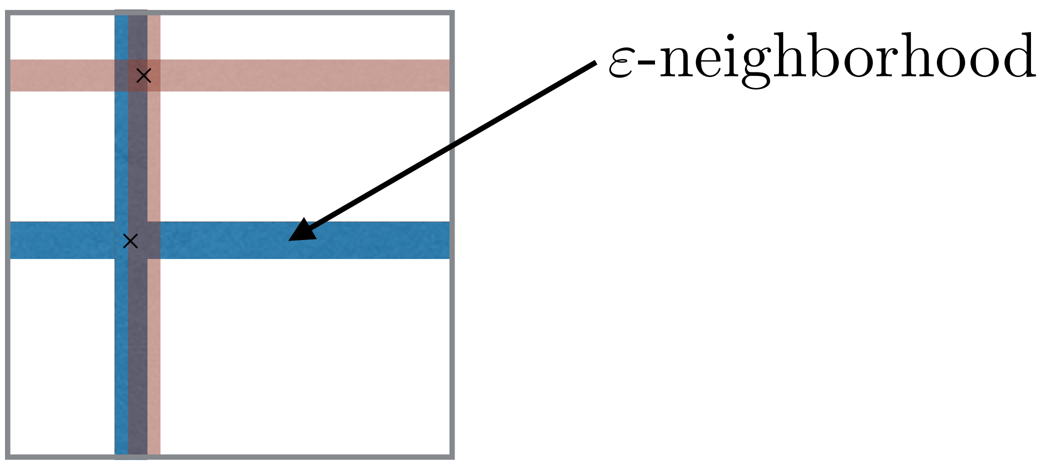
Model 2: Distance

- For each pair (i,j), we *independently* connect v_i and v_j with prob.

$$p_{i,j} = p(w_i, w_j, x_i, x_j) = \Theta\left(\min\left\{1, \left(\mathrm{Vol}(B_{i,j})^{-1} \cdot \frac{w_i w_j}{n}\right)^{\alpha}\right\}\right),$$

where $\alpha > 1$ is a parameter,
and $B_{i,j}$ is the ball around x_i with radius $d(x_i, x_j)$.

Example: minimum component distance



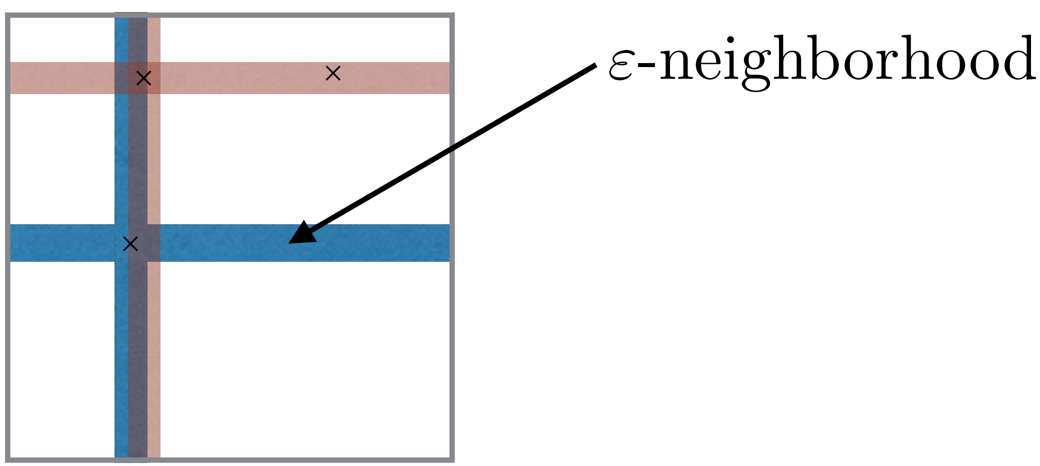
Model 2: Distance

- For each pair (i,j), we *independently* connect v_i and v_j with prob.

$$p_{i,j} = p(w_i, w_j, x_i, x_j) = \Theta\left(\min\left\{1, \left(\mathrm{Vol}(B_{i,j})^{-1} \cdot \frac{w_i w_j}{n}\right)^{\alpha}\right\}\right),$$

where $\alpha > 1$ is a parameter,
and $B_{i,j}$ is the ball around x_i with radius $d(x_i, x_j)$.

Example: minimum component distance



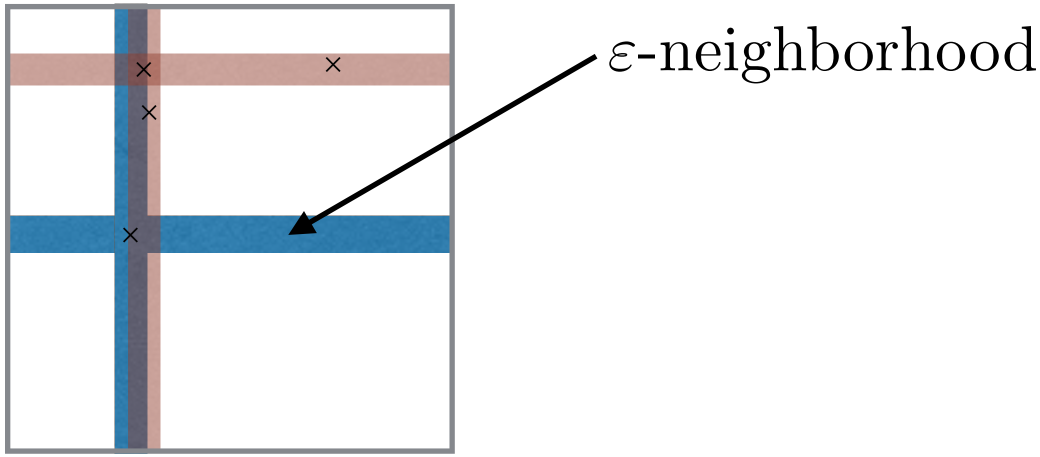
Model 2: Distance

- For each pair (i,j), we *independently* connect v_i and v_j with prob.

$$p_{i,j} = p(w_i, w_j, x_i, x_j) = \Theta\left(\min\left\{1, \left(\mathrm{Vol}(B_{i,j})^{-1} \cdot \frac{w_i w_j}{n}\right)^{\alpha}\right\}\right),$$

where $\alpha > 1$ is a parameter,
and $B_{i,j}$ is the ball around x_i with radius $d(x_i, x_j)$.

Example: minimum component distance



Summary

General Model:
- power law degrees
- small world: components, diameter, average distance

Distance Model:
- strong clustering (if distance function is “nice”)
- may be non-rigid clustering

Euclidean Model (or other norms):
- small separators
- small entropy, efficient compression
- linear time sampling

Future Work

Algorithms
- communication protocols
- de-anonymization

Processes
- infection processes (work in progress)
- information dissemination

Others
- recovering the underlying geometry
- attacks
- dynamic graph problems
- games on graphs

Thank you for your attention!

Questions?