Generating Networks with Realistic Properties Based on a Given (Set of) Network(s)

And a Short Overview of the KONECT Project

Jérôme Kunegis

Université de Namur, 2016-12-06

WeST People and Knowledge Networks

Reveal

Institute for Web Science and Technologies · University of Koblenz-Landau, Germany

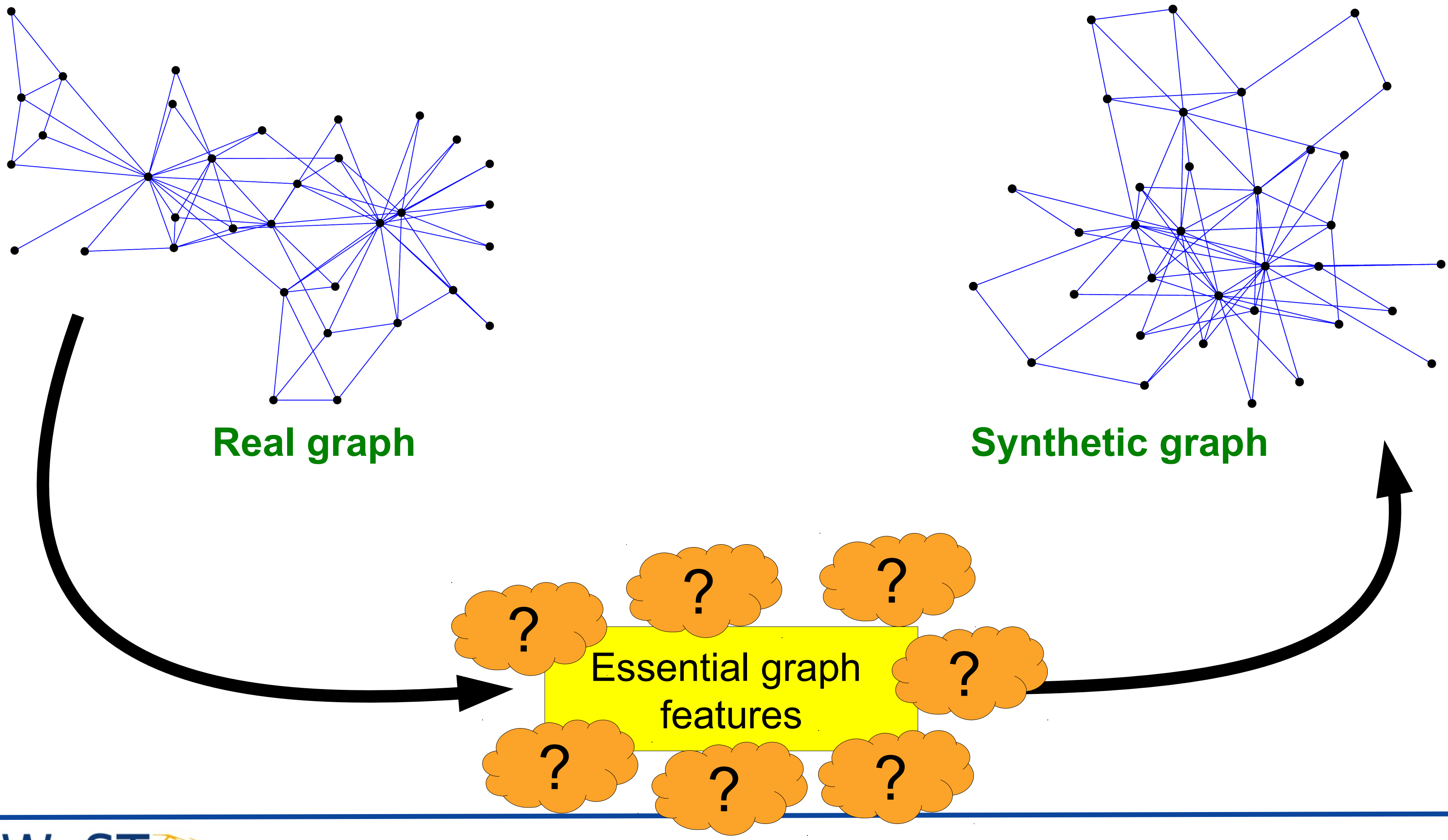
Outline

Part I: Network Models

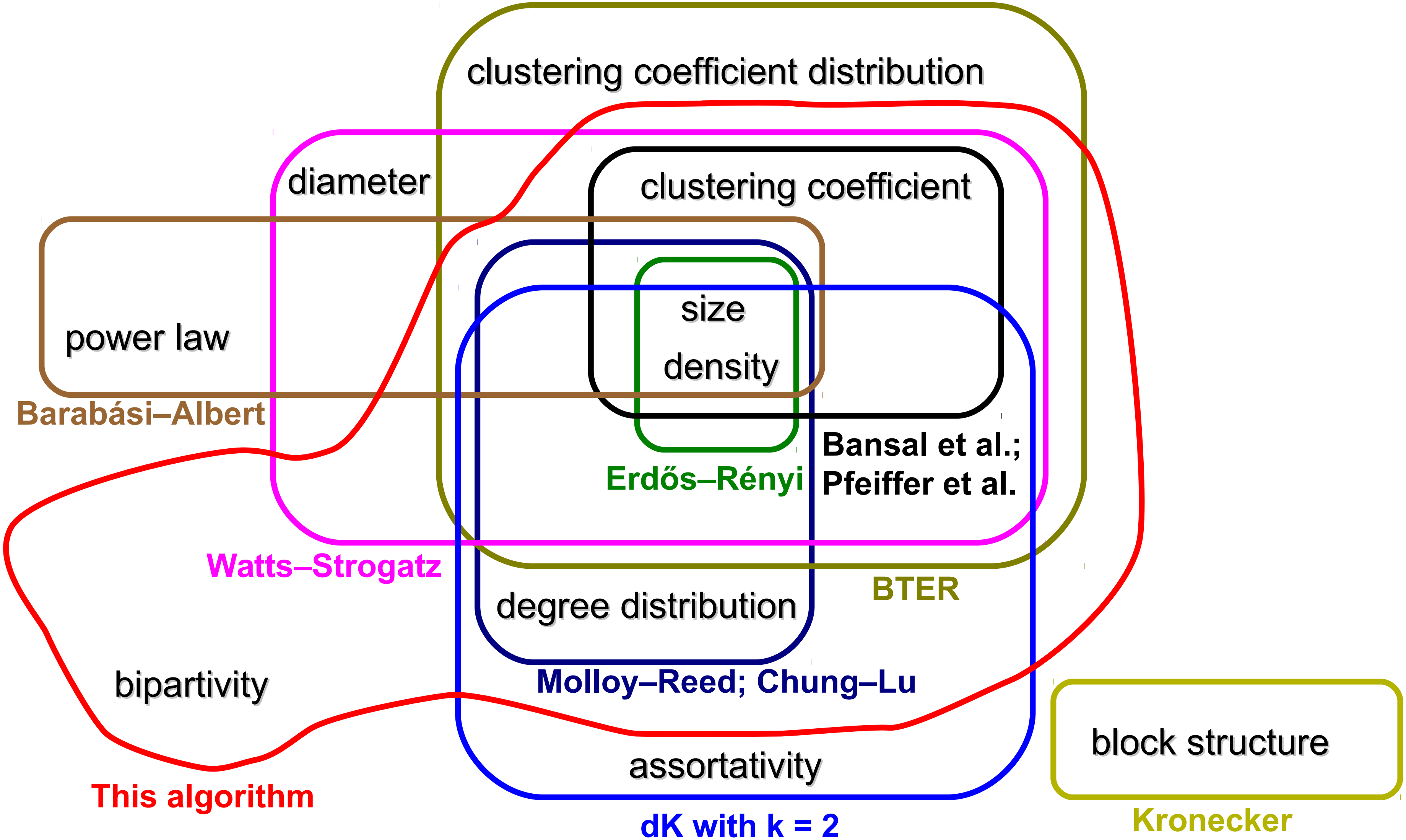
Part II: Network Set Models

Part III: KONECT

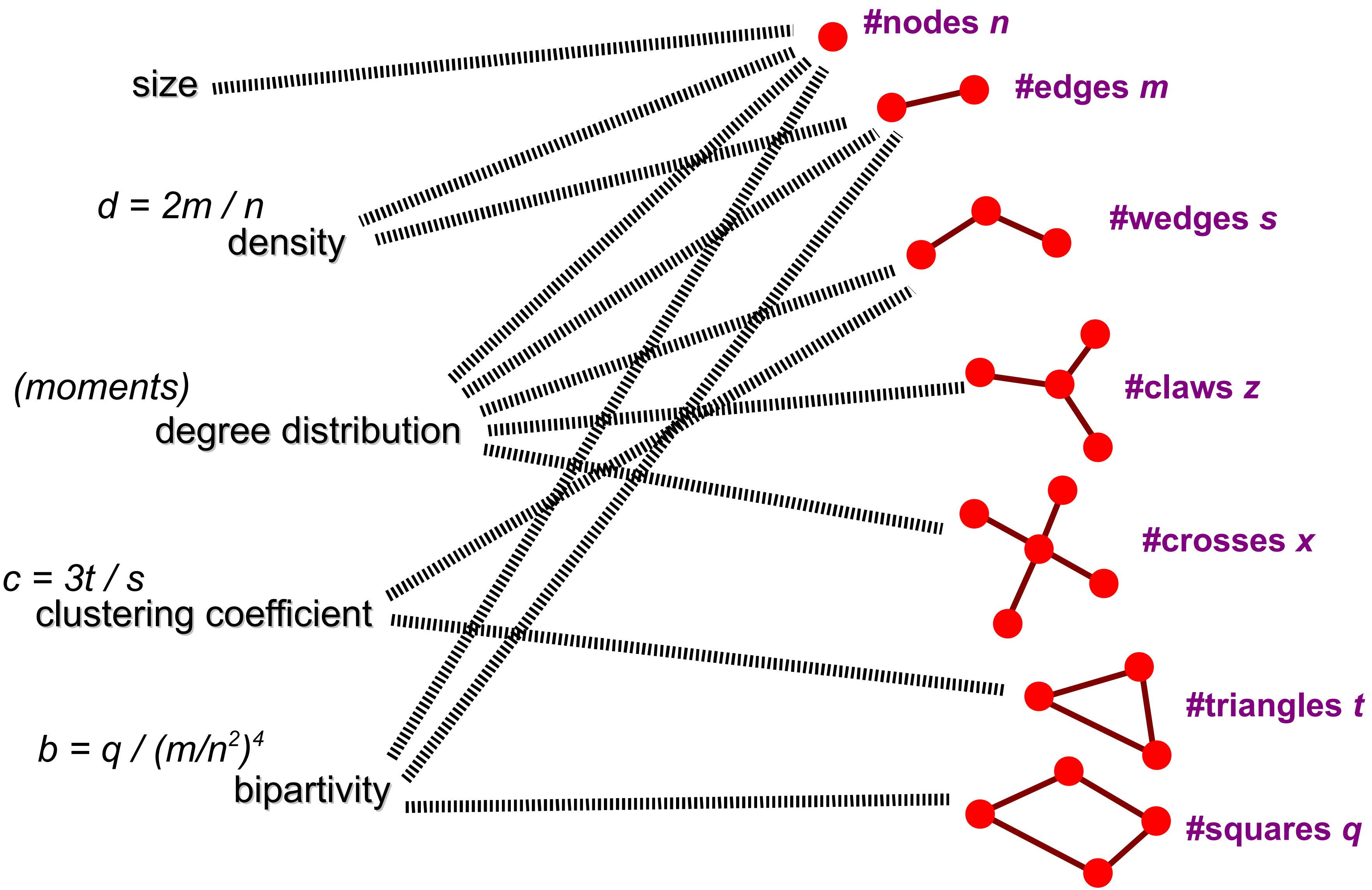
Part I: Network Models



Essential Features



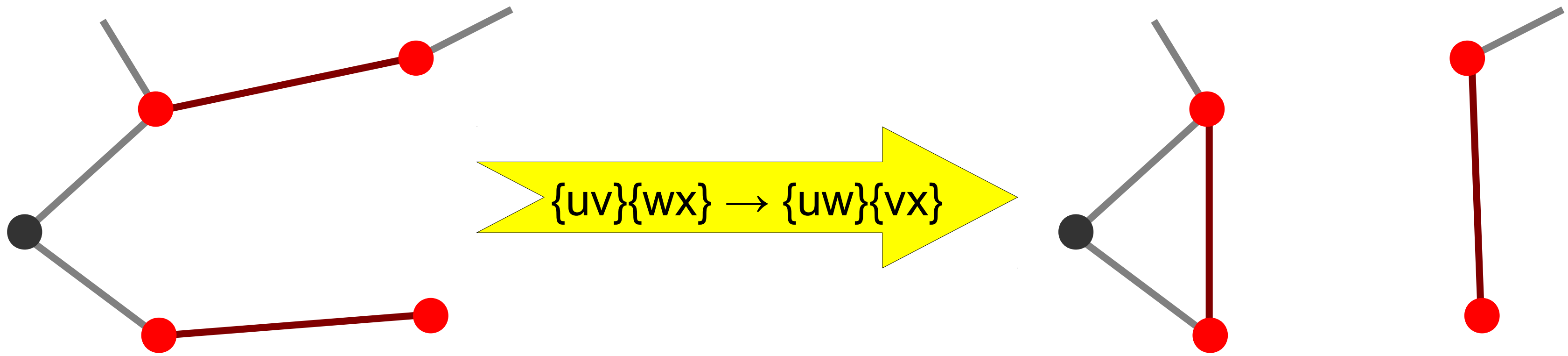
Mapping Features to Statistics



Generating a Graph with Given Subgraph Counts

Traditional approach:

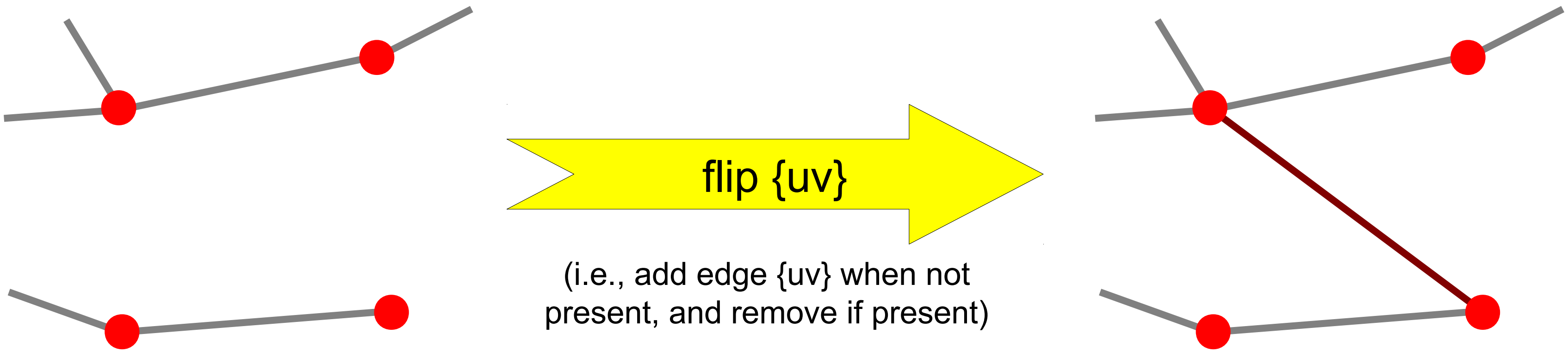
- Generate graph with number of nodes and edges (Erdős–Rényi)
- Generate graph with given degree distribution (Molloy–Reed)
- Perform “switches” to adjust the number of triangles (Pfeiffer et al.) (Bansal et al.)



- Task: find a switch that maintains n, m, t and changes q

Too difficult

Idea: All Statistics Are Variable



Does the flip make the graph better?

⇒ We need a measure of distance to the requested values

$$E = \left[\frac{1}{|\mathcal{S}|} \sum_{S \in \mathcal{S}} \underbrace{\left(\frac{S(G) - S(G_0)}{S(G_0)} \right)}_{\text{relative error } E_S}{}^{2} \right]^{1/2}$$

$\mathcal{S}$: count statistics

G: current graph

G_0: requested graph

Algorithm – Basic Version

```
G = ErdősRényi(n, p)

repeat {
    Choose node u at random
    for each node w ≠ u {
        E_w = E(G ± {uw})
    }
    v = argmin_w≠u E_w
    G = G ± {uv}
} until E has not reached a new minimum value in the
  last (−n ln ε) iterations
```

Slow!

because E is recomputed for each node w

Make sure (1 − ε) of all nodes are visited.
We use ε = 0.01

Notation: G ± {uv} is G with {uv} flipped

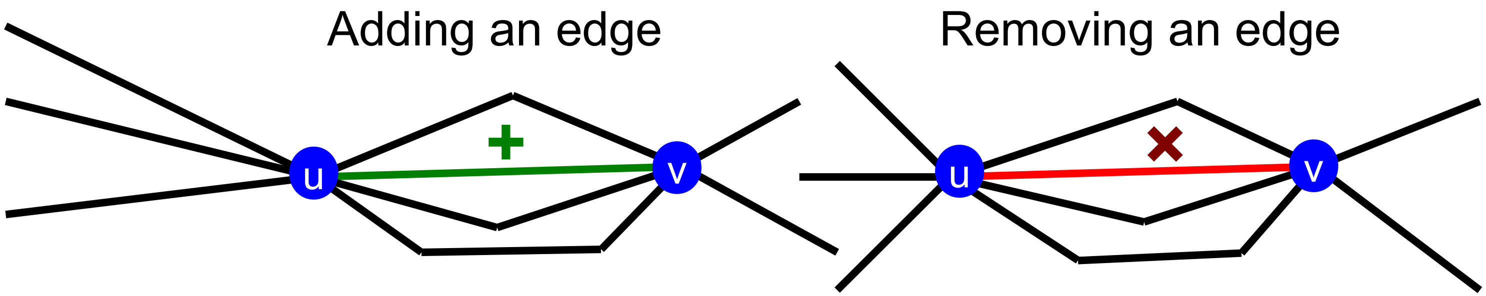
Algorithm – Exploit Updatability of Count Statistics

```
G = ErdősRényi(n, p)
for all statistics S {
    x_S = S(G)
    y_S = S(G_0)
}

repeat {
    Choose node u at random
    for each node w ≠ u {
        for all statistics S {
            Δ_S_w = Diff(G, {uw}, S)
        }
        E_w = sum_S ((y_S + Δ_S_w − x_S) / x_S)^2
    }
    v = argmin_w≠u E_w
    G = G ± {uv}
    for all statistics S { y_S = y_s + Δ_S_w }
} until  E has not reached a new minimum value in the
  last (−n ln ε) iterations
```

Compute the change in statistic S when flipping {uv} in G

Implementing Diff(G, {uv}, S)



	Adding an edge	Removing an edge
Diff(G, {uv}, #edges) =	+1	−1
Diff(G, {uv}, #wedges) =	d(u) + d(v)	−d(u) − d(v) + 2
Diff(G, {uv}, #claws) =	(d(u) choose 2) + (d(v) choose 2)	−(d(u)−1 choose 2)−(d(v)−1 choose 2)
Diff(G, {uv}, #crosses) =	(d(u) choose 3) + (d(v) choose 3)	−(d(u)−1 choose 3)−(d(v)−1 choose 3)
Diff(G, {uv}, #triangles) =	CommonNeighbors(u,v)	−CommonNeighbors(u,v)
Diff(G, {uv}, #squares) =	Walk_3(u,v)	−Walk_3(u,v) + d(u) + d(v) − 1

Algorithm – Vectorized Version

```
G = ErdősRényi(n, p)
for all statistics S {
    x_S = S(G)
    y_S = S(G_0)
}

repeat {
    Choose node u at random
    for all statistics S {
        Δ_S = Diffvect(G, u, S)
    }
    E = sum_S ((y_S + Δ_S − x_S) / x_S)^2
    v = argmin_w≠u E_w
    G = G ± {uv}
    for all statistics S { y_S = y_s + Δ_S_w }
} until  E has not reached a new minimum value in the
  last (−n ln ε) iterations
```

Compute changes of statistic S for flips between u and all other nodes

Implementing Diffvect(G, u, S)

Vectorized operations and data:

$\mathbf{A}$	Adjacency matrix of G
$\mathbf{A}_{:u}$	Adjacency vector of u in G
$\mathbf{d}$	Degree vector of G
$\mathbf{x} \circ \mathbf{y}$	Component-wise product of vectors

Runtime for each iteration: O(n)
Number of iterations: O(n)
Total runtime: $O(n^2)$

$$\text{Diffvect}(G, u, \#\text{edges}) = -2\,\mathbf{A}_{:u} + 1$$

$$\text{Diffvect}(G, u, \#\text{wedges}) = (-2\,\mathbf{A}_{:u} + 1) \circ (\mathbf{d} + d(u)) + 2\,\mathbf{A}_{:u}$$

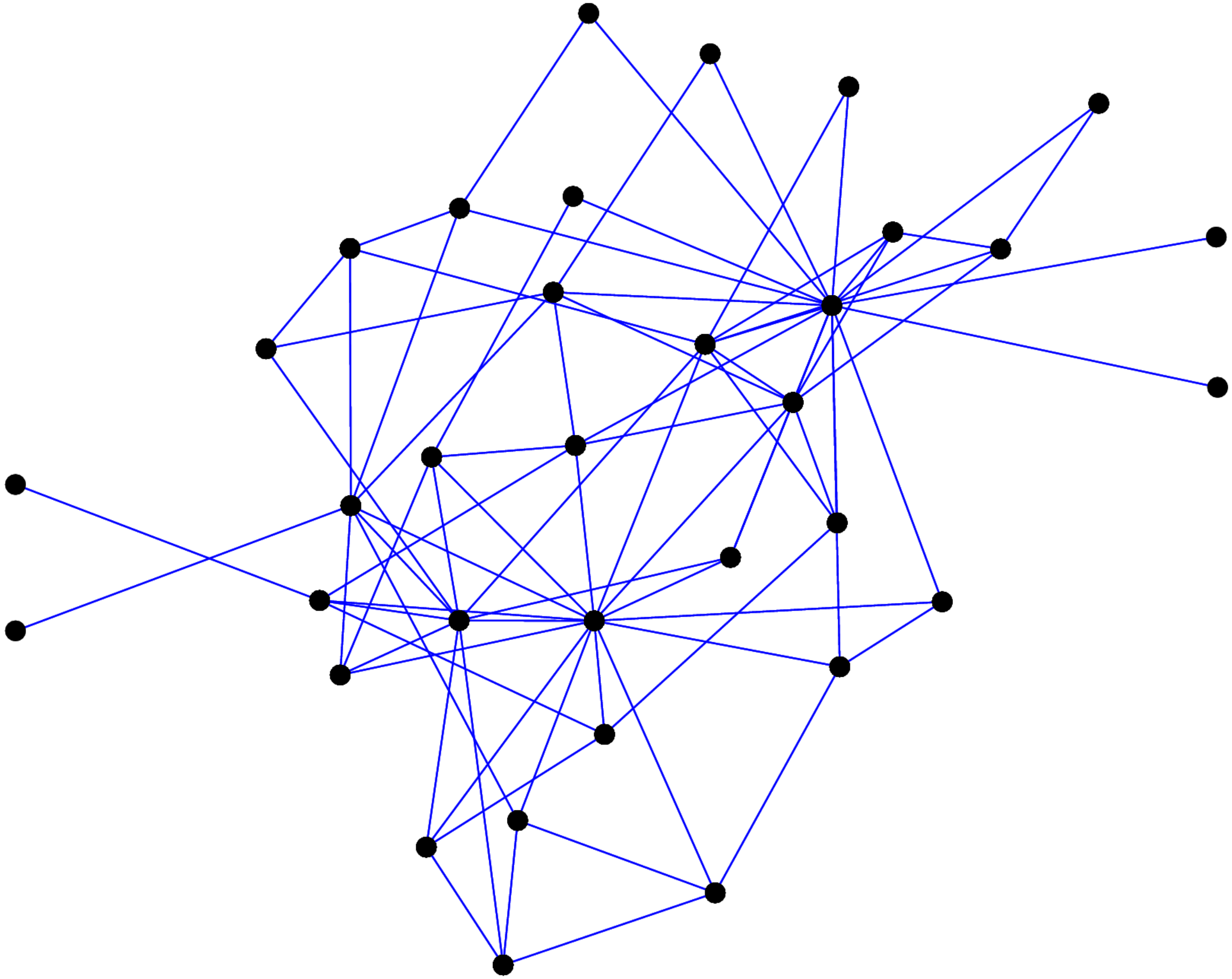
$$\text{Diffvect}(G, u, \#\text{claws}) = (1/2)\,(-2\,\mathbf{A}_{:u} + 1) \circ [(\mathbf{d} - \mathbf{A}_{:u}) \circ (\mathbf{d} - 1 - \mathbf{A}_{:u}) + (-\mathbf{A}_{:u} + d(u)) \circ (-\mathbf{A}_{:u} - 1 + d(u))]$$

$$\text{Diffvect}(G, u, \#\text{crosses}) = (1/6)\,[(\mathbf{d} - 1) \circ (\mathbf{d} - 2) \circ (\mathbf{A}_{:u} \circ (-2\,\mathbf{d} + 3) + \mathbf{d}) + (d(u) - 1)(d(u) - 2)((3 - 2\,d(u)\,\mathbf{A}_{:u} + d(u))]$$

$$\text{Diffvect}(G, u, \#\text{triangles}) = (\mathbf{A}\,\mathbf{A}_{:u}) \circ (-2\,\mathbf{A}_{:u} + 1)$$

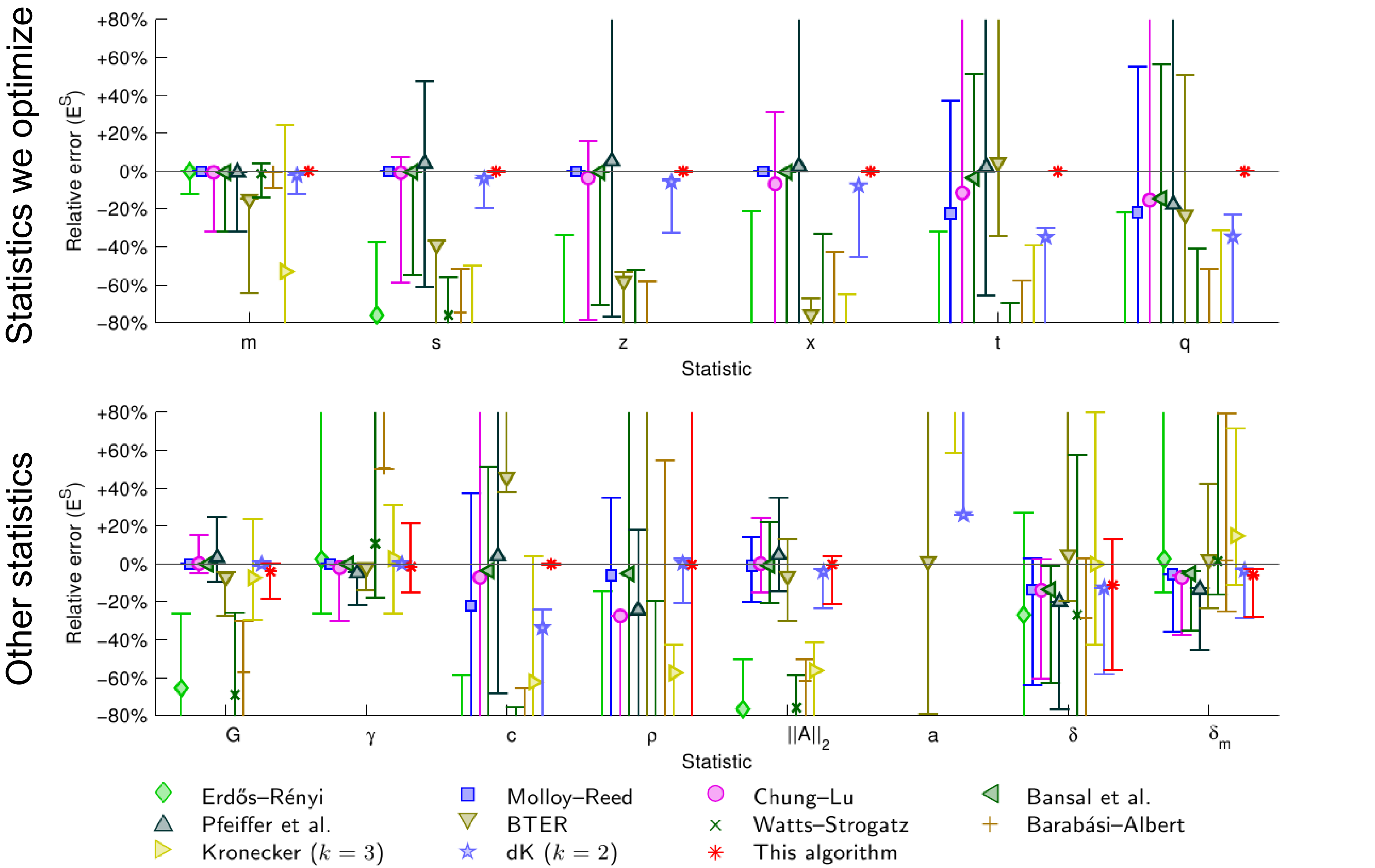
$$\text{Diffvect}(G, u, \#\text{squares}) = (\mathbf{A}^2\,\mathbf{A}_{:u}) \circ (-2\,\mathbf{A}_{:u} + 1) + \mathbf{A}_{:u} \circ (\mathbf{d} + d(u) - 1)$$

Generate Network with Same Properties as Zachary's Karate Club



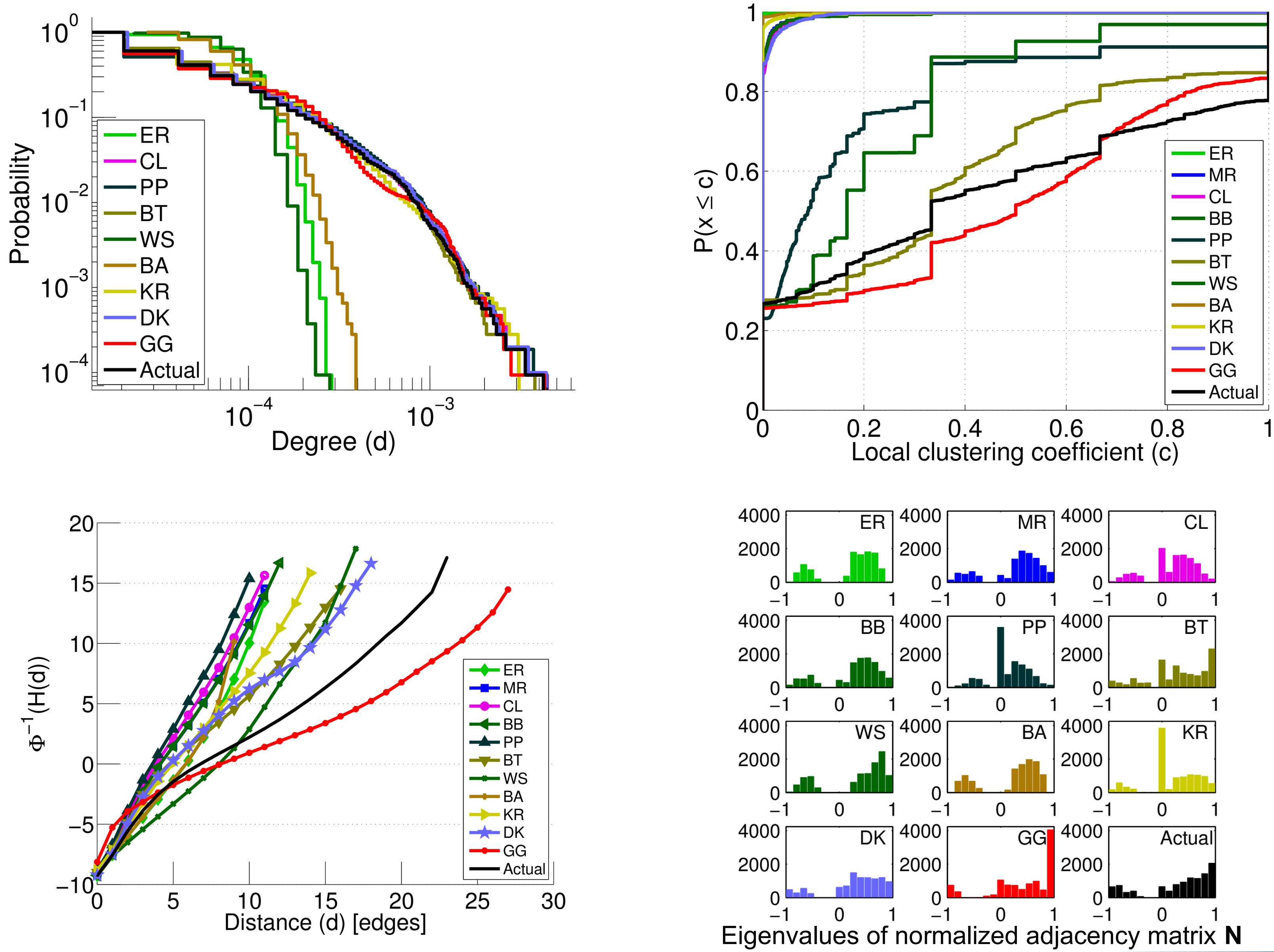
Experiment: Precision (all datasets)

Plots show median and 10th/90th percentiles over 36 networks

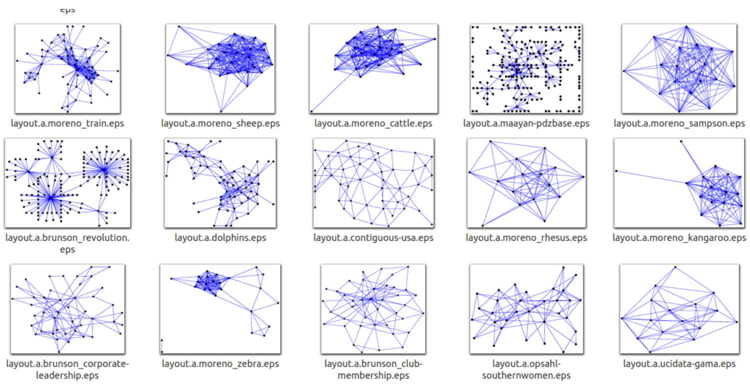


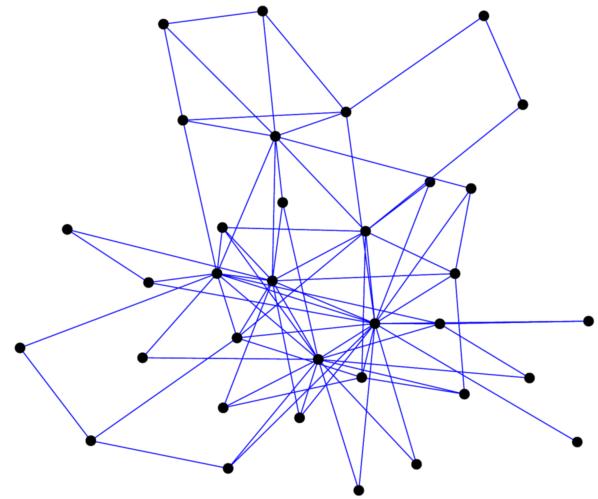
Qualitative Experiments

(Pretty Good Privacy network)



Part II: Network Set Models



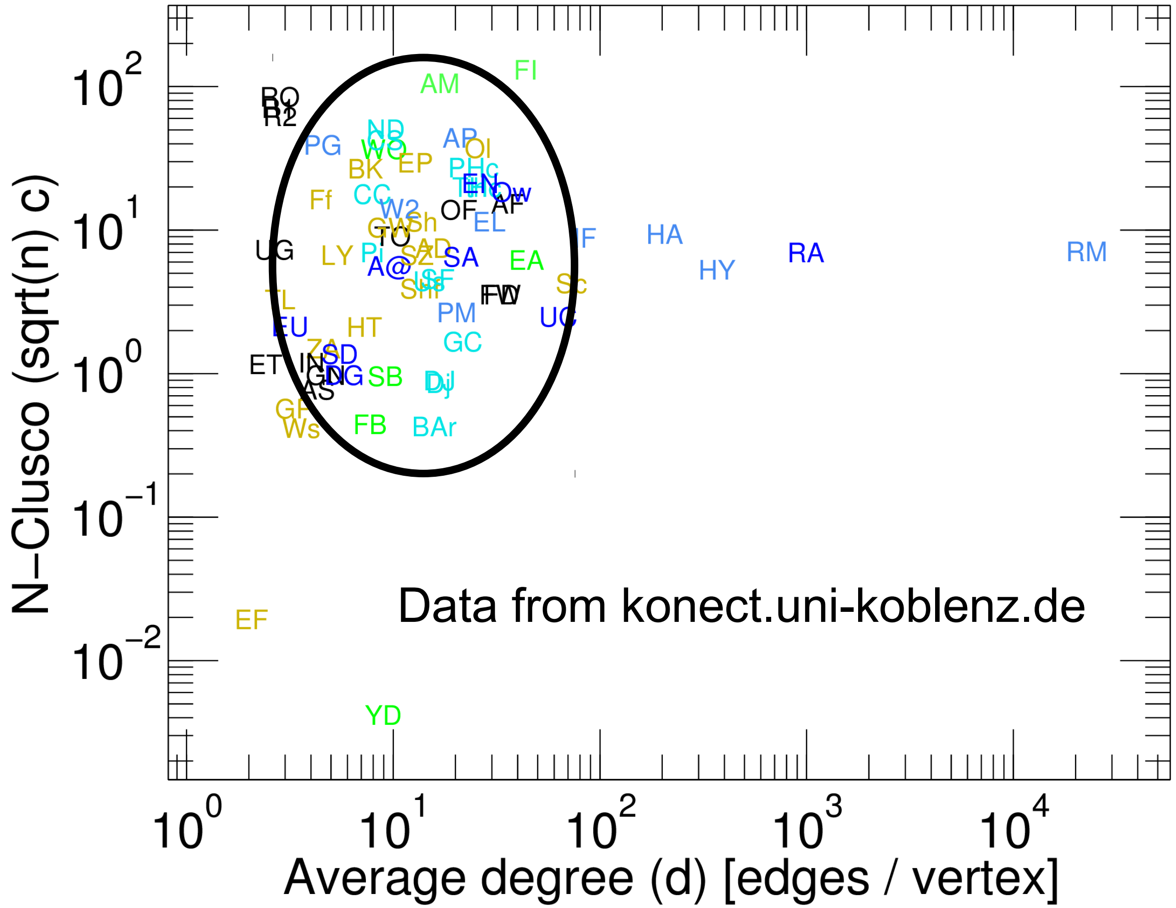


Real graphs

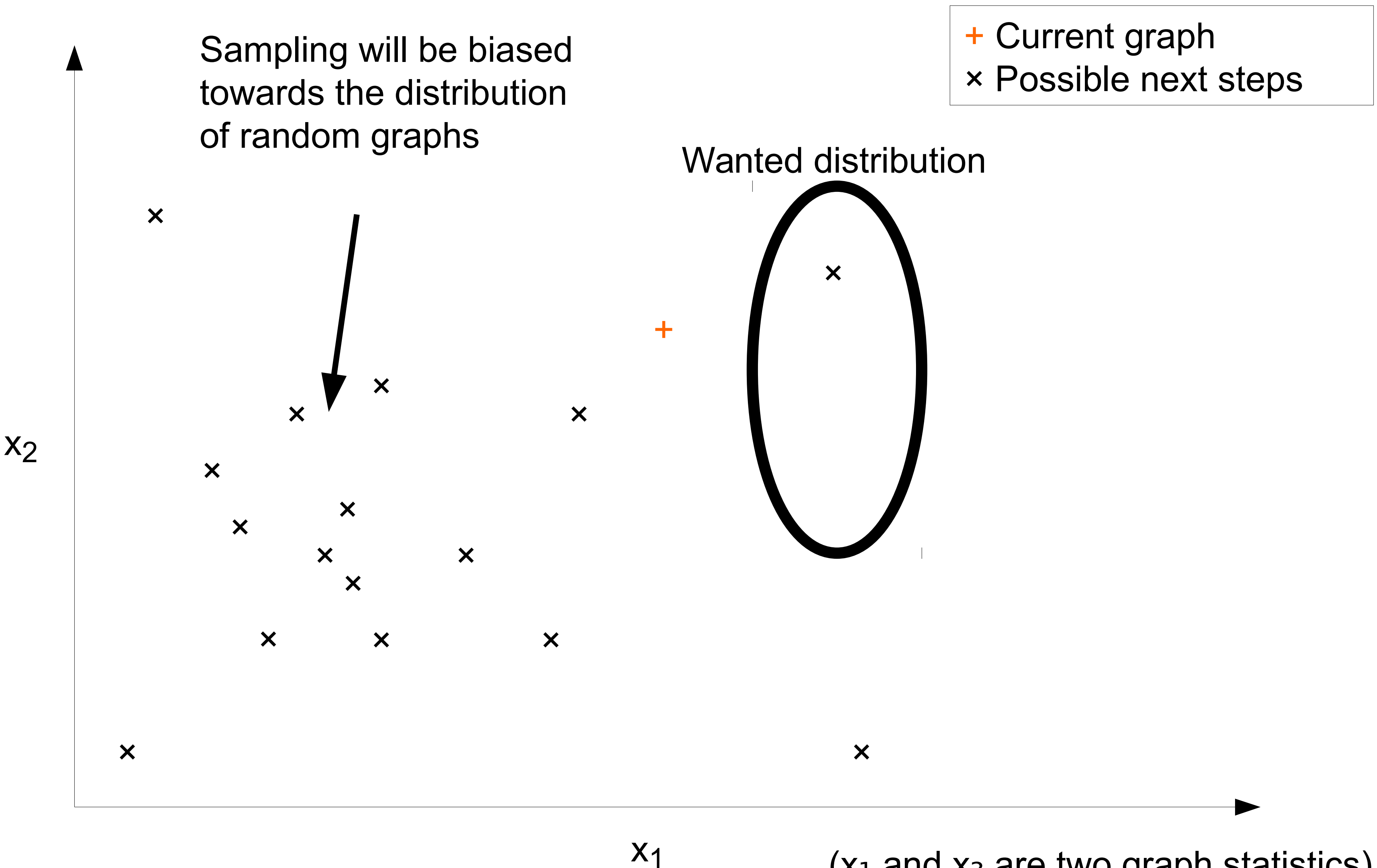
Synthetic graph

Essential graph features

Real Networks Have a Distribution of Values

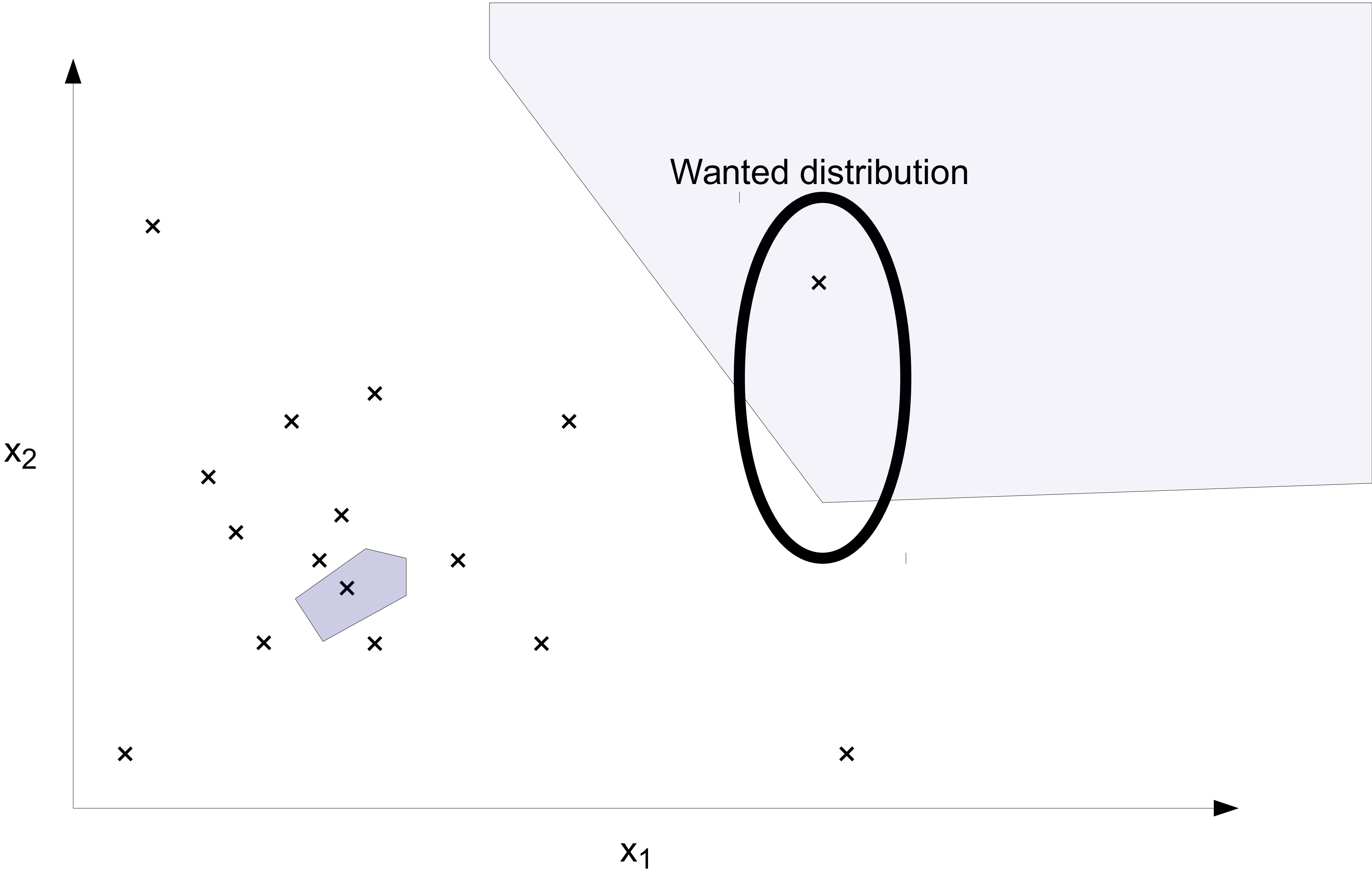


Monte Carlo Markov Chain Methods



(x1 and x2 are two graph statistics)

Solution: Integral of Measure of Voronoi Cells



How to Compute the Integral over Voronoi Cells

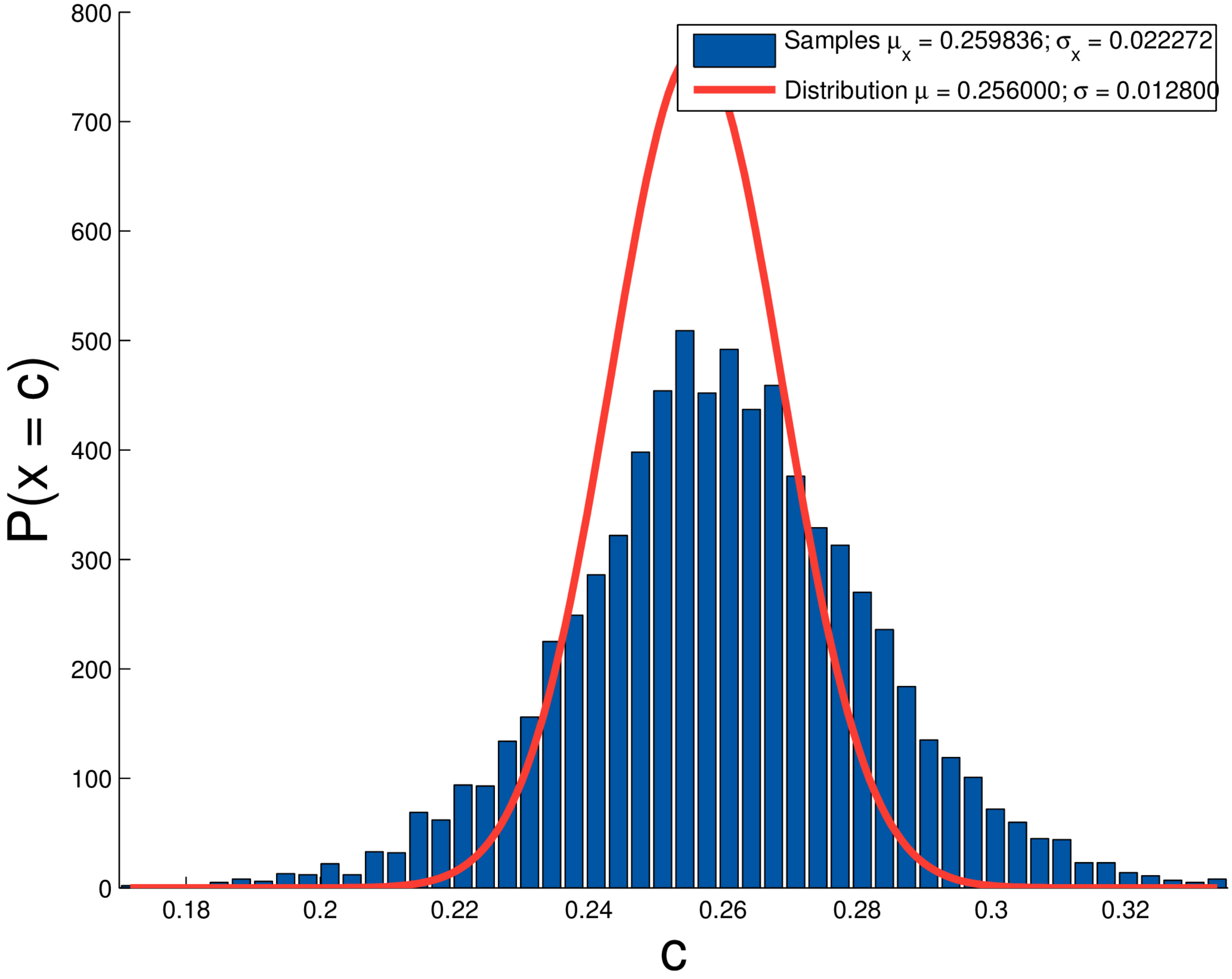
Answer: We don't have to.

Sampling strategy:

- Sample point in statistic-space according to our wanted distribution
- Find nearest possible network (i.e., nearest “×”)

Claim: This distribution at each step is similar to the underlying measure, giving an unbiased sampling.

Result: Close, But Not Exact

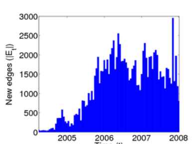


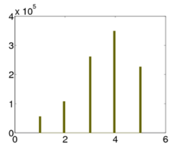
Part III: KONECT.uni-koblenz.de

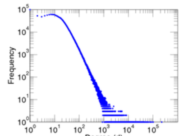
KONECT – Koblenz Network Collection

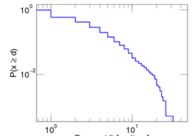
Code	Name ▲	Category	F.	W.	M.	n	m	c
CL	Actor collaborations	Misc	U	=		382,219	33,115,812	16.6%
ME	Adolescent health	HumanSocial	D	+		2,539	12,969	14.2%
AD	Advogato	Social	D	+		6,541	51,127	9.22%
TC	Air traffic control	Infrastructure	D	−		1,226	2,615	6.39%
CA	Amazon (MDS)	Misc	U	−		334,863	925,872	20.5%
Am	Amazon (TWEB)	Misc	D	−		403,394	3,387,388	16.6%
AP	arXiv astro-ph	Coauthorship	U	−		18,771	198,050	31.8%
PH	arXiv hep-ph	Coauthorship	U	−		28,093	4,596,803	28.0%
PHc	arXiv hep-ph	Citation	D	−		34,546	421,578	14.6%
TH	arXiv hep-th	Coauthorship	U	−		22,908	2,673,133	26.9%
THc	arXiv hep-th	Citation	D	−		27,770	352,807	12.0%
BAi	Baidu internal	Hyperlink	D	=		2,141,300	17,794,839	0.245%
BAr	Baidu related	Hyperlink	D	=		415,641	3,284,387	0.0663%
BS	Berkeley/Stanford	Hyperlink	D	−		685,230	7,600,595	0.694%
MB	Bison	Animal	D	+		26	314	78.9%
Mg	Blogs	Hyperlink	D	−		1,490	2,220,035	100%
BK	Brightkite	Social	U	−		58,228	214,078	11.1%
PM	Caenorhabditis elegans	Metabolic	U	=		453	4,596	12.4%
IN	CAIDA	Computer	U	−		26,475	53,381	0.732%

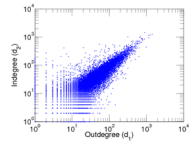
Square count	$q = \lvert\{u,v,w,x \mid u \sim v \sim w \sim x \sim u\}\rvert / 8$
4-tour count	$T_4 = 8q + 4s + 2m$
Power law exponent	$\gamma = 1 + n \left(\sum_{u \in V} \ln \frac{d(u)}{d_{\min}} \right)^{-1}$
Gini coefficient	$G = \frac{2 \sum_{i=1}^{n} i d_i}{n \sum_{i-1}^{n} d_i} - \frac{n+1}{n}$
Relative edge distribution entropy	$H_{\mathrm{er}} = \frac{1}{\ln \lvert V \rvert} \sum_{u \in V} - \frac{d(u)}{D} \ln \frac{d(u)}{D}$

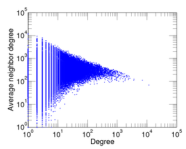

Temporal distribution

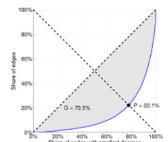

Edge weight and multiplicity distribution

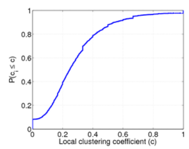

Degree distribution

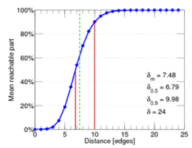

Cumulative degree distribution

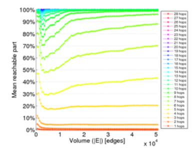

Out/indegree comparison

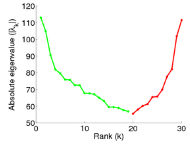

Assortativity plot

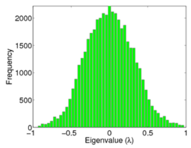

Lorenz curve

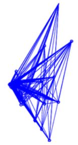

Clustering coefficient distribution

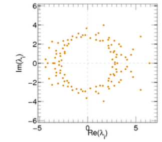

Distance distribution

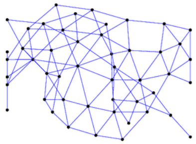

Temporal distance distribution

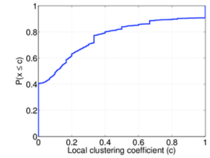

Top-k eigenvalues

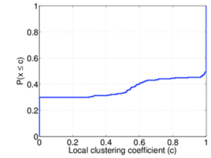

Spectral plots

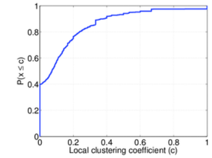

Network visualization

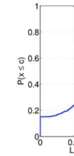

Complex eigenvalues of the asymmetric adjacency matrix

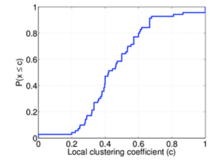

Layout

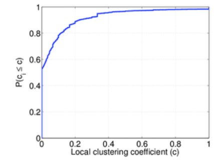

Gowalla

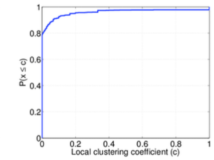

Haggle

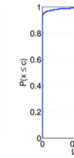

Hamsterster friendships

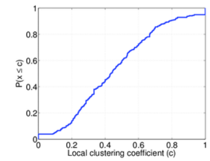

H

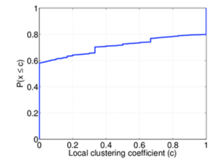

Highschool

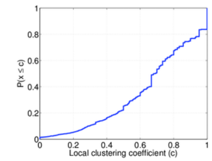

Hudong internal links

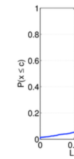

Human protein (Figeys)


Hum


Infectious


Internet topology


JDK dependency


JUNG ar

KONECT.uni-koblenz.de

258 network datasets as of December 2016

undirected / directed / bipartite, unweighted / multiple edges / signed / ratings / etc., loops, timestamps

32 categories: social, rating, text, contact, lexical, interaction, infrastructure, hyperlink, computer, citation, authorship, animal, etc.

All code on GitHub: https://github.com/kunegis

KONECT Analysis with Stu

```
%CPU     Mem [k]      Runtime  Runtime left Log
  40       55068  1-13:49:30              /data/kunegis/tmp/ifub.lasagne-yahoo
  41      751920    18:45:53              /data/kunegis/tmp/julia.kunegis.inter2.log.wiki_talk_zh
  42      863196     9:54:10              /data/kunegis/tmp/julia.kunegis.inter2.log.wiki_talk_es
  43      567820     8:31:25              /data/kunegis/tmp/julia.kunegis.inter2.log.bibsonomy-2ui
  43      687976     7:38:18              /data/kunegis/tmp/julia.kunegis.inter2.log.bibsonomy-2ti
  42     2094952     5:21:49 173-10:17:18 /data/kunegis/tmp/m.kunegis.hopdistr_time_comp.full.munmun_twitterex_ti.log
  43      741520     5:15:42              /data/kunegis/tmp/julia.kunegis.inter2.log.munmun_twitterex_ti
  44     1982928     4:26:56   3-23:27:11 /data/kunegis/tmp/m.kunegis.cluscod.youtube-links.log
  47     2119532     3:10:17  10-16:11:33 /data/kunegis/tmp/m.kunegis.cluscod.wiki-Talk.log
  49     1701260     2:46:27   4-11:24:27 /data/kunegis/tmp/m.kunegis.statistic_comp.squares.wiki-Talk.log
  48     1580524     2:23:17   4-17:09:06 /data/kunegis/tmp/m.kunegis.statistic_comp.tour4.wiki-Talk.log
  48     2032020     2:11:45  14-07:50:19 /data/kunegis/tmp/m.kunegis.hopdistr_time_comp.full.digg-votes.log
  48      905720     2:10:22              /data/kunegis/tmp/julia.kunegis.inter2.log.digg-votes
  50     2060064       35:48      6:54:45 /data/kunegis/tmp/m.kunegis.cluscod.petster-cat-friend.log
  49     1466944       30:44      2:58:58 /data/kunegis/tmp/m.kunegis.statistic_comp.squares.petster-cat-friend.log
  50     1570328       24:46      2:37:59 /data/kunegis/tmp/m.kunegis.statistic_comp.tour4.petster-cat-friend.log
```

Stu Example from KONECT

```
#
# Degree vs local clustering coefficient
#

@degcc:   [dat/dep.degcc];

>dat/dep.degcc:  dat/NETWORKS_SQUARE
{
        for network in $(cat dat/NETWORKS_SQUARE) ; do
                echo @degcc."$network"
        done
}

@degcc.$network:  plot/degcc.a.$network.eps;

plot/degcc.a.$network.eps:
        m/degcc.m $[-t MATLABPATH]
        dat/cluscod.$network.mat
        uni/out.$network
{
        ./matlab m/degcc.m
}
```

Thank You

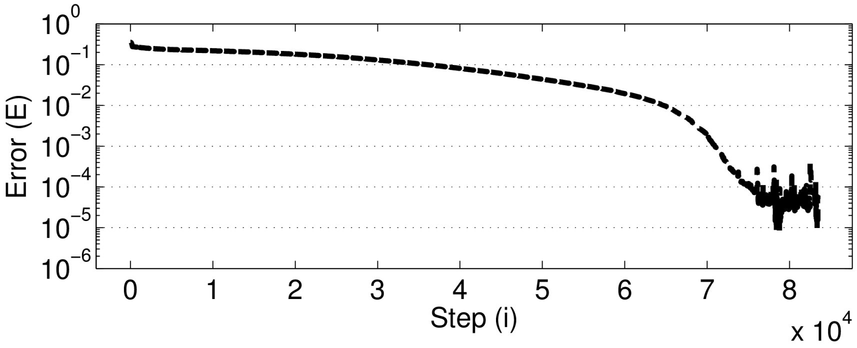
All data available at the Koblenz Network Collection (KONECT):
http://konect.uni-koblenz.de/

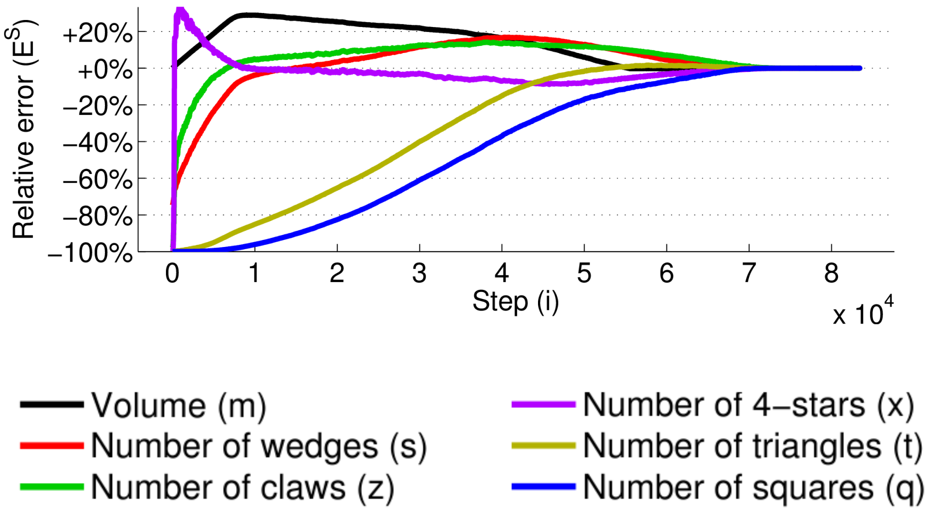
→ Network contributions accepted ←

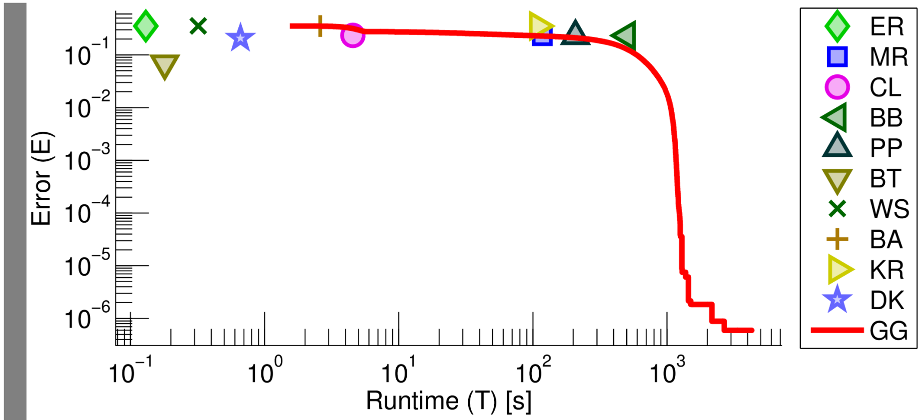
Stu: https://github.com/kunegis/stu

@kunegis

Experiment: Convergence







(Pretty Good Privacy network)

Experiment: Scalability