Monte Carlo Generation of Equilibrated Graphs

Hannes Nagel[1], Piotr Białas[2], Bartłomiej Wacław[1] and Wolfhard Janke[1]
[1]Institut für Theoretische Physik, Universität Leipzig, Germany
[2]Institute of Computer Science, Jagellonian University Kraków, Poland

Abstract

We propose a C++ library for generating and handling random graphs with given statistical weights. The modular and extendable set of functions allows the user to easily create a program that generates complex networks with prescribed node-degree distribution, node-node correlations and assumed global structure (trees, simple graphs or degenerated graphs), with no a-priori limitation on the size of graph. The library also contains functions to perform statistical estimations on graphs or to export the graphs for further external processing or visualization.

Introduction

Random networks can be described in the framework of statistical physics by introducing an ensemble of random graphs. This approach allows one to use standard methods, e.g., to calculate various observables as ensemble averages. The space of graph shapes is the same as in the Erdősz-Rényi model of random graphs, but in addition every graph in the ensemble has a certain statistical weight $W(\alpha)$ which tells what is the probability of its occurence.
The weight $W(\alpha)$ enhances the probability of some graphs, for example those with some special properties. As a result, "typical" graphs may have a power-law degree distribution, desired degree-degree correlations, small diameter, large number of triangles or any other desired feature.
In the simplest case, the weight $W(\alpha)$ is a product of local graph properties, as node degrees:

$$W(\alpha) = \prod_{i=1}^{L} p(k_i),$$

which gives the possibility of tuning the degree distribution by adjusting the function $p(k)$ appropriately.
A more generic but still local weight

$$W(\alpha) = \prod_{\langle i,j \rangle} p(k_i, k_j),$$

allows one to introduce degree-degree correlations between adjacent nodes.

Method

- Propose a local modification

  $$\text{graph } \alpha \rightarrow \text{graph } \beta$$

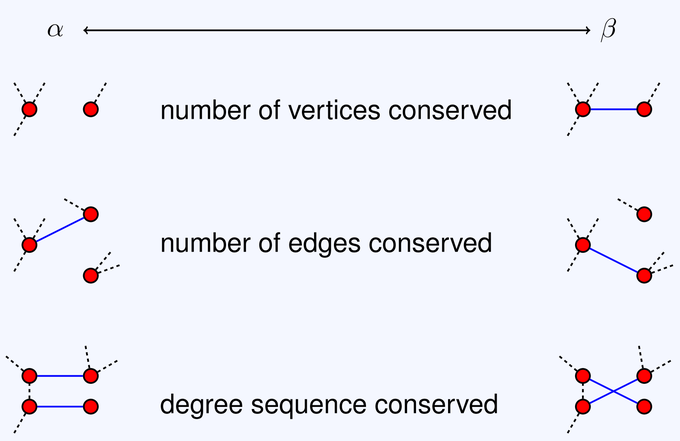
  The type of modification proposed depends on which global property (simple, multi- or tree graph) or conservation law (conservation of links or degree sequence) is assumed.

- Metropolis: Accept β with probability P_{acc} or reject:

  $$P_{\text{acc}} = \min\left\{1, \frac{W(\beta)}{W(\alpha)}\right\}$$

- Calculate quantities of interest as ensemble averages from the sampled graphs

Local modifications conserving global graph properties



Examples

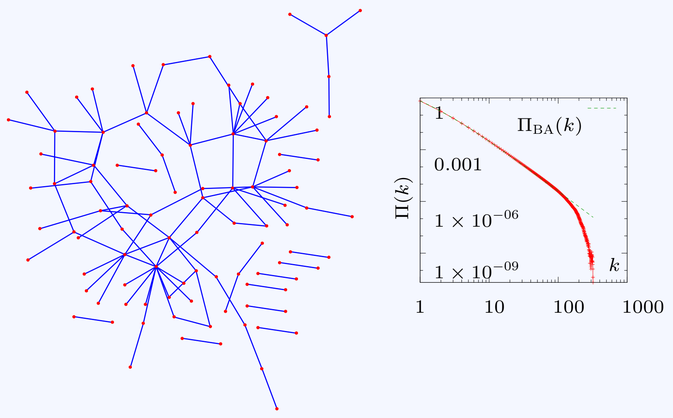
- Equilibrated network with Barabási-Albert degree distribution

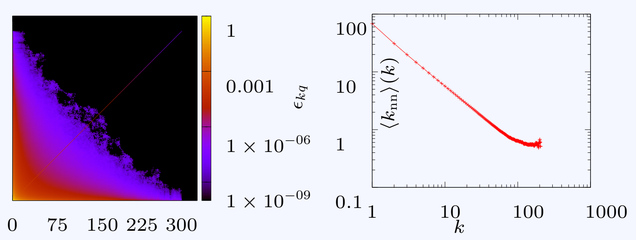
  $$\Pi_{\text{BA}} = \frac{4}{k(k+1)(k+2)}$$

  shows disassortativity in $k_{\text{nn}}(k)$. The weight

  $$p(q) = \frac{q!}{q(q+1)(q+2)}$$

  gives the desired degree distribution and low degree-degree correlations (see *lower left*).

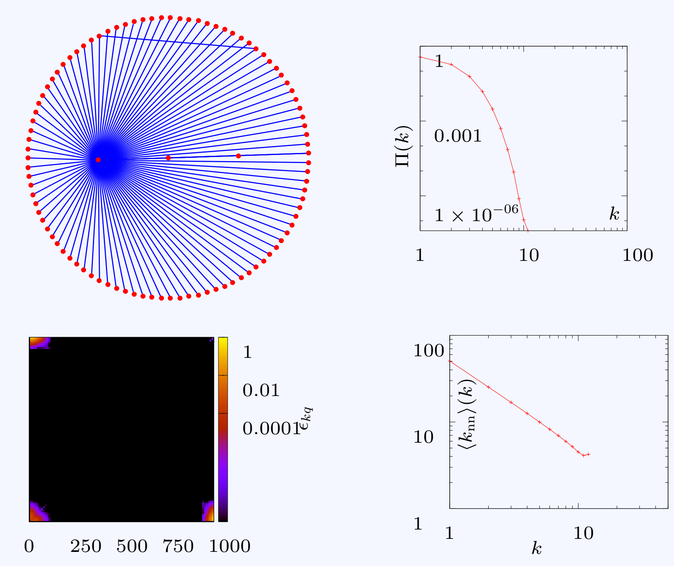




- Introduction of explicit degree-degree correlations

  $$p(k,q) = |q-k|^2 \ ,$$

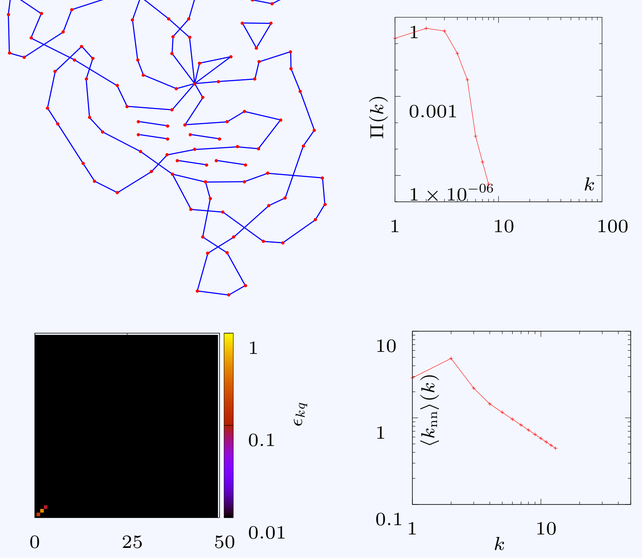
  which prefer links that connect high degree sites to low degree sites



  Degree correlations are much stronger now: Links connecting low degree sites with high degree sites are far more probable than others.

- Correlations with preference for connections of sites with similar degree

  $$p(k,q) = \frac{1}{|q-k|^2 + 1}$$

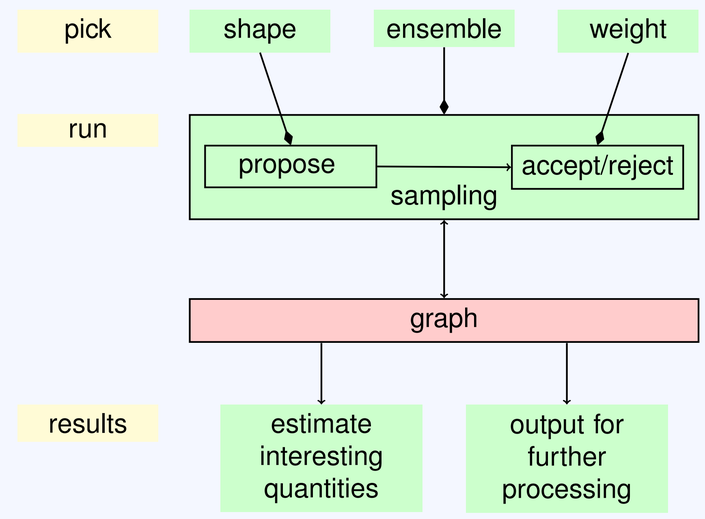


Simulations: simple graphs with $N, L = 1000$. Pictures of graphs: $N = 100, L = 100$

Minimal Example

```
#include<fstream>
#include"graphgen.hpp"

double p(unsigned int q){
  if(q>0) return q*(1.0+q)/(q+3.0); else return 1e20;
}

using namespace graphgen;
typedef undirected_graph Graph;
typedef vertex_weight<value_type::ratio> Weight;
typedef canonical<Weight, shape::simple> Ensemble;

const int NV=1000;

int main(){
  Graph graph(NV);
  graph.gen_BarabasiAlbert(1);

  Weight weight(p);
  Ensemble E( graph, weight );
  degreedist Deg(2*NV);

  for( int n=0; n<1000000; n++ ){
    E.sweep(NV);
    Deg.measure(graph);
  }

  std::ofstream degdstr("degree");
  degdstr<<Deg;
  std::ofstream gstream("graphs");
  gstream<<graphgen::graphviz(graph)<<std::endl;
}
```

Method Scheme



Extendability

The modularity of the program and the usage of C++ template classes allows one to extend the code easily, in order to simulate more complicated graph ensembles, for instance:

- non local functional weights
- other classes of functional weights (e.g. not necessarily degree-dependant)
- sampling of graphs with special global shapes
- "your demand here"

References

General Complex Networks

- S. N. Dorogovtsev, J. F. F. Mendes, A. N. Samukhin, *Principles of statistical mechanics of random networks*, Nucl. Phys. B **666** (2003) 396.

Graph Ensemble Sampling

- L. Bogacz, Z. Burda, and B. Wacław, *Homogeneous complex networks*, Physica A **366** (2006), 587.
- L. Bogacz, W. Janke, and B. Wacław, *A program generating random graphswith given weights*, Comp. Phys. Comm. **173** (2005) 162.

GraphGen Package

- Documentation and download **`http://www.physik.uni-leipzig.de/~nagel/graphgen`**.

Gedruckt im Universitätsrechenzentrum Leipzig