Escola d'Enginyeria de Telecomunicació i Aeroespacial de Castelldefels
UNIVERSITAT POLITÈCNICA DE CATALUNYA

TREBALL FINAL DE GRAU

TÍTOL DEL TFG: Cascade failures in complex networks

TITULACIÓ: Grau en Enginyeria d'Aeronavegació

AUTOR: Llorenç Sánchez Marín

DIRECTOR: Francesc Comellas Padró

DATA: 30 de Maig de 2018

Overview

Globalization plays an important role in our society. There is a need to be connected and have the ability to reach any place in the world, for this purpose different networks or systems had been created such as: Internet, transportation and communication systems, the power grid, social networks etc., which can be classified as complex networks and each of them is specialized in a need.

These networks should have a high resilience against possible failures in any of its nodes. That can occur either randomly or because of planned attacks. Against this failures, the network should be able to redirect their flow through other nodes that are still operative.

Recent studies have considered models for node cascade failures that lead to a network collapse as it has happened recently in real life networks like Twitter, Facebook, the global router network and the European electrical network.

In this TFG we analyse the behaviour of different networks properties (random, small-world, scale-free, modular, geometrical, etc.) when they are confronted to node cascade failures. We study which network invariants and parameters affect the process.

We have performed simulations and analysed the results by using the Python's package "NetworkX", which allows the creation, manipulation and analysis of complex networks.

To perform the analysis we have compared the obtained behaviours according to the network structure and the nodes' functions. Afterwards, we have used the data from a real-life network like the USA air transport network and we have compared it with the behaviour of the generated networks. By this, the research shows us that what makes a network to collapse is mainly its structure instead of its centrality, and to reduce the network collapsing risk it is necessary that all nodes are well-connected among them at a local level.

Resumen

La globalización juega un papel importante en la sociedad actual. Existe la necesidad de estar conectado y tener la capacidad de llegar a cualquier lugar del mundo, para ello se han creado diversas redes o sistemas como: Internet, los sistemas de transporte y comunicación, la red eléctrica, las redes sociales, etc., que pueden ser catalogadas como redes complejas y cada una está especializada en una necesidad.

Estas redes deben tener una alta resiliencia frente a posibles fallos en alguno de sus nodos. Esto puede ocurrir tanto de manera aleatoria como por un ataque planeado. Frente a estos fallos, la red debe ser capaz de redirigir el flujo a través de otros nodos aún operativos.

Estudios recientes han obtenido modelos acerca de la caída de nodos en cascada que consiguen un colapso de la red como ha ocurrido recientemente en redes reales como Twitter, Facebook, la red global de rúters o la red eléctrica europea.

En este TFG analizamos el comportamiento de diferentes propiedades en las redes (aleatorias, mundo-pequeño, escala libre, modulares, geométricas, etc.) cuando se enfrentan a una caída en cascada de los nodos. Estudiamos que parámetros y variantes de la red afectan al proceso.

Hemos realizado las simulaciones y analizado los resultados mediante el paquete “NetworkX” de Python, permite la creación, manipulación y análisis de redes complejas.

Para realizar el análisis hemos comparado los comportamientos obtenidos según la estructura de la red y la función de los nodos. Después, hemos utilizado los datos de una red real como la red de transporte aéreo de los EUA y los hemos comparado con el comportamiento de las redes generadas. Con esto hemos podido ver que lo que hace fallar a una red mayoritariamente es su estructura, y que para reducir el riesgo de colapso es necesario que los nodos estén bien interconectados entre ellos a nivel local.

INDEX

0 Introduction

Many real complex systems such as communication, transport, social, or biological networks are “small-world scale-free” and quite sensitive to cascade failures. It is really important to prevent networks from failures that let the whole network or a huge part of it unserviceable. This TFG aims to find efficient methods for network protection. For this purposes, we will study model graphs and networks coming from real data, such as the USA airport network, in order to analyse graph invariants and parameters which are relevant in cascade failures.

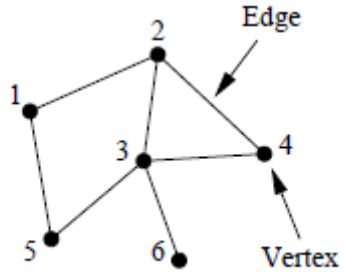
During the last two decades, there have been important advances in the context of network science. A relevant study by Albert-Jeong-Barabasi [23] concludes that real networks, such as the USA power grid, are robust against random attacks, but weak against planned attacks. Researchers further conclude that it is possible a complete failure of the whole network just by the failure of few nodes [23]. The aim of this project is to analyse the results and behaviour of several network families after a cascade failure and identify the parameters that affect it.

To carry out this project we have generated several network families with different properties each. On these generated networks, its nodes, have a different initial load depending on the parameter considered (centrality) and thus, a different maximum load value that the node can accept before failing. If one of the nodes fails (due to an attack, due an error or due to any incident that might occur in real life) transfers their load to their neighbours (nodes to whom they are connected). This extra load can also make the neighbours fail to produce a cascade failure. The behaviour of the network during these cascade failures is what we are considering in our study. For the purpose of this study, we have selected three different scenarios: when the network fails after a failure in nodes with the highest load, with the lowest load and with a half value of the highest load. We show that network failures are mostly affected by the nature of the network rather than by the centrality considered. Likewise, the failures are primarily affected by how well every individual node is connected with the other nodes around them (its neighbourhood) instead of how well is connected with the rest of the nodes of the whole network.

To support our research, we have considered the USA air transport network. After forcing a network failure we have analysed the role of different centralities and compared the behaviour with those of the simulated networks. The results obtained from this real network are similar to the ones found with the generated graphs.

1 Graph Theory

A graph is a representation of a system. It consists of nodes (vertices), which represent the entities of the system and links (edges) which connect a pair of nodes and represent a particular kind of interconnection between those entities. Networks are modelled by using graph theory which can describe mathematical concepts and represent the essential topological properties of a network by treating it as a collection of nodes and edges.



Mathematically, a graph $G = (V, E)$ is a pair of sets of vertices (V) and edges (E).

1.1.Basic Glossary

- Two vertices are *adjacent* if there is an edge which links them.
- The *order (n)* of a graph is the number of vertices it has: $n = |V|$.
- The *size (m)* of the graph is the number of edges it has: $m = |E|$.
- The *degree* of a vertex i (k_i) is the number of edges attached to it.
  In many cases, the vertices with the highest degrees in a network, those with the most connections, also called hubs and play an important role in the system.
- The *degree* of a graph (Δ) is the highest degree of all nodes: $\Delta = max\ (k_i)$.
- A *path* is the succession of edges that link two vertices of the same graph.
- The *length* of the path is defined by the number of edges that this path crosses.
- A *geodesic path* between a pair of vertices is the shortest path that links them, and it gives also, by definition, the *distance (d)* between them. It can exist more than one geodesic path between two nodes.
- The *eccentricity* of a vertex v (e_v) is the maximum distance between that vertex v to all other vertices in the graph: $e_v = max\{d(v,x)\}$
- The *radius (r)* of a graph is the minimum eccentricity of a graph. A node is called central if its eccentricity is equal to the radius of the network.

1.2.Graph Parameters and Properties

There exist many parameters and measures that could characterize different topological and dynamical properties of a network. For this project, we will focus only on the ones that will give us a useful information for our analysis.

1.2.1 Diameter

The diameter *(D)* of a graph is the length of the longest geodesic path between any pair of vertices in the network for which a path actually exists, in other words, it is the maximum eccentricity of a graph.

The diameter is restricted to:

$$r \leq D \leq 2r \quad \textbf{(1.1)}$$

Where r is the radius if the graph.

The diameter of a graph could be affected substantially by a small change to only a single vertex or a few vertices, which makes it a poor indicator of the behaviour of the network as a whole.

1.2.2 Mean Distance

The mean distance of a graph is defined as the average of the distance between all pairs of vertices:

$$\bar{d} = \frac{\sum\sum d(u,v)}{n(n-1)} \quad \textbf{(1.2)}$$

1.2.3 Wiener Index

The Wiener index is a topological index defined as the sum of lengths of the shortest paths, $d(u, v)$, between all pair of nodes of a network.

$$W(G) = \frac{1}{2}\sum_{u}\sum_{v} d(u, v) \quad \textbf{(1.3)}$$

Notice that $$\bar{d} = \frac{2W(G)}{n(n-1)} \quad \textbf{(1.4)}$$

You can estimate the overall structure of the graph if it is dense (high index) or sparse (low index). [2]

1.2.4 Clustering Coefficient

The clustering coefficient of a graph is a measure of the degree to which its vertices tend to cluster together, i.e. if neighbours of a vertex are also neighbours among them.

It is related to the number of triangles in the graph. The clustering is high if two vertices sharing a neighbour have a high probability of being connected to each other.

The clustering of a node i is the fraction of possible triangles through that node that exists in the graph:

$$C_i = \frac{2|C_3(i)|}{k_i(k_i - 1)} \tag{1.5}$$

Where $C_3(i)$ is the number of triangles connected to the node i.

Then the clustering coefficient for the graph is the average:

$$C = \frac{1}{n}\sum_i C_i \tag{1.6}$$

Figure 1 *Example of a simple network with one triangle and eight connected triples*

1.2.5 Degree distribution

The degree distribution of a graph is the probability distribution of all degrees over the graph.

For a random graph, the degree distribution follows a Poisson distribution:

$$p(k) = \frac{e^{-\bar{k}}\bar{k}^k}{k!} \tag{1.7}$$

In real networks with a high number of vertices we usually find a power law distribution:

$$p(k) = Ak^{-\gamma} \tag{1.8}$$

In a power-law distribution, the probability of finding a node with degree k decreases as a negative power of the degree: $p(k) \sim k^{-\gamma}$. This means that the probability of finding a high-degree node is relatively small in comparison with a high probability of finding low-degree nodes.

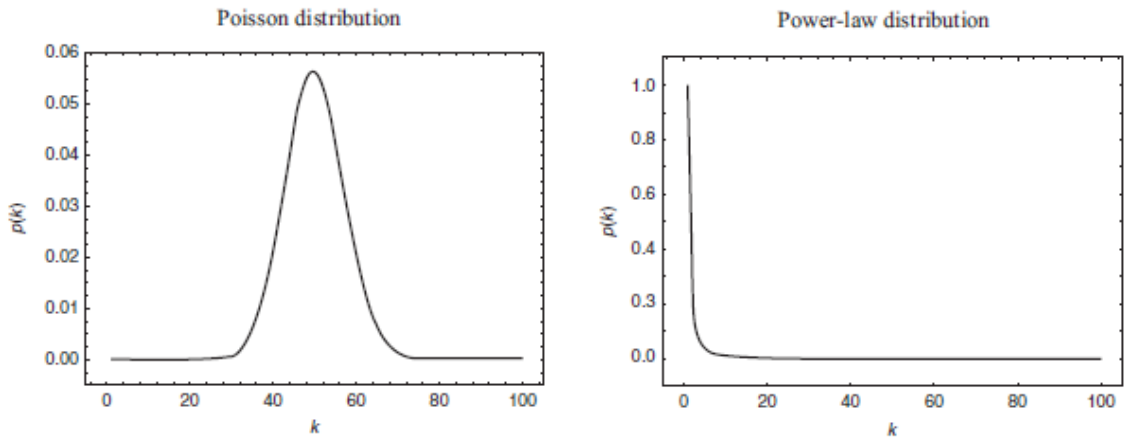


Illustration 1 *Examples of typical degree distributions.*

The networks that present this kind of behaviour are called scale-free networks. Most real networks present a power-law distribution because in every real network there are nodes that are more important than others and they have a high number of edges connected to them.

1.2.6 Centrality

Centrality is the property that shows us the importance of a node inside the whole network in comparison with the other nodes, such as its closeness to many other nodes or the number of times a node is included in a geodesic path of a pair of nodes, on their way to reach others as an intermediary.

Quantifies how important vertices (or edges) are and determine many of the structural and functional properties of this node in a networked system, for example, how important a person is within a social network or how well-used a road is within an urban network. It is based on counting paths going through a node. For each node, *i*, in the network, the number of routing paths to all other nodes going through *i* is counted, and this number determines the centrality of the node *i*.

Social network analysts, in particular, have expended considerable effort studying it. There are a wide variety of mathematical measures of vertex centrality that focus on different concepts and definitions of what it means to be central in a network.

There are different measures of centrality that are used in network analysis but we will just focus on six different centralities. These are explained in section ***2.2***.

1.2.7 Small-World Effect

Watts and Strogatz observed that many real networks, apparently random, have some common properties such as small diameter, in comparison with random networks with the same order and size, but also a high clustering coefficient whilst in random networks is around 0.

This led Watts and Strogatz to introduce the concept of “small-world” network. They modelled them from an ordered lattice, such as the *k*-ring, a ring where each site is connected to its 2*k* nearest neighbours of both sides (see *Figure 2*). For each node, each of the links going out from it is removed with probability ϕ and is rewired to a randomly selected site in the network. A variant of this process is to add links rather than rewire, which simplifies the analysis without considerably affecting the results.

The nodes with this random links are called “hubs”. These hubs are responsible for the small-world phenomenon, they reduce the distance among all nodes.

Real networks seem they have a chaotic network, however it is found empirically that the mean distance, between nodes, is very short (can be defined as $log(n)$) relative to a highly ordered network. [1]

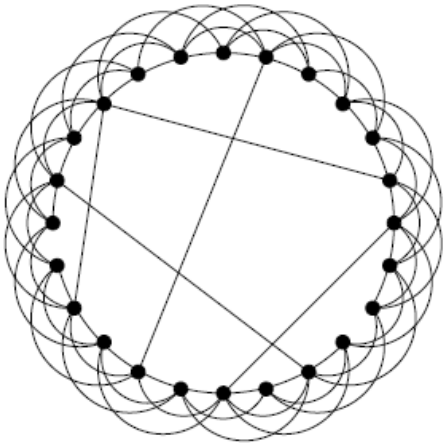


Figure 2 *Example of creation of a small-world network*

In popular culture, this effect is referred to the “six degrees of separation” concept. This name comes from the experiment that the psychologist Stanley Milgram did sending out letters to certain people in what he considered a remote city (in Ohio). Each letter would name a Milgram’s friend living in Boston and asked the recipient to forward the message directly to that person, or, if they didn't know the person, to forward the message to a direct contact that they thought might know the end person. In one study, out of 296 letters, only about 64 reached their destination. Many original and intermediate recipients ignored the request. When people did continue the chain, and the letter reached its destination to Boston, Milgram and his collaborators counted the number of hops it made from the initial recipient to its target. After several experiments, they concluded that it takes an average of 6 people to connect any two persons.

1.2.8 Robustness

The robustness of a network is the ability to keep most of its relevant properties and characteristics from the initial scenario against a random failure of any node. It has been shown that scale-free networks are robust against random errors but vulnerable to planned attacks.

In random networks, where the degree distribution follows a Poisson law, random and planned errors affect in a similar way, only with few node failures the connectivity between active nodes gets worse and it produces a network failure.

Research about real-world networks is very important, they can be used to protect lots of infrastructures based on networks, and thus protect the crucial nodes efficiently to avoid a network collapse.

1.2.9 Modularity

Modularity is a measure of networks' structure. Many complex networks consist of a number of modules or groups, so modularity was designed to measure the strength of division of the network. Each module contains several interconnected nodes, and there are few connections between nodes in different modules.

Networks with high modularity have dense connections between the nodes in the same module but few connections between nodes in different modules. Inside the modules, we can differentiate between two different kinds of nodes in terms of their roles inside their community structure: Provincial hubs, which are connected mainly to nodes in their own modules; and Connector hubs, which are connected to nodes from other modules. [19]

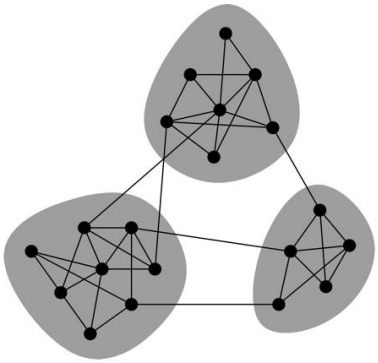


Figure 3. *Vertices in many networks fall naturally into groups or communities creating a modular network.*

1.2.10 Connection density

The connection density parameter is a proportion of the number of edges in the network and the total number of possible edges. It is the simplest estimator of the physical cost of a network. A high density of connections between regions increases the clustering coefficient of the graph, whereas the long-range connections between different modules or clusters, even though they are relatively few in number, keep the path length low. [20]

Examples of high connection density networks are: an epidemic disease spreading, the neurological network of the brain or the telecommunication network.

The connection density parameter (D) can be computed as:

$$D = \frac{2 \cdot m}{N(N-1)} \quad \textbf{(1.9)}$$

Where m is the number of edges and N the number of nodes. For a network of N nodes. The maximum value of the connection density parameter is 1.

2 Graphs and Centralities

2.1.Graphs

For our study we have chosen a variety of random and real networks:

2.1.1 Geographical Threshold 2D and 3D

This kind of graphs analyse the geographical extension of the non-growing scale-free networks. Even the Internet, could seem it is not restricted to physical distances because signals travel at the speed of light, but the Internet is subjected to geographical constraints due to wiring costs.

Real networks are often represented as topological spaces. It is often advantageous to map non-physical quantities or networks into geographical spaces by, for example, the principal component analysis

Weights represent the fitness of vertices to win edges and are interpreted as, for example, capitals, social skills, activity levels, information contents, concentration or mass of physical or chemical substances, and the vertex degree itself.

The geographical threshold graph model places n nodes uniformly at random in a rectangular domain. Each node u has assigned a weight (w) w_u. Two nodes u and v are joined by an edge if and only if:

$$w_u + w_v \geq \theta \cdot r^{\alpha} \qquad \textbf{(2.1)}$$

Where r is the Euclidean distance between u and v, θ the threshold value and “α” a distance parameter.

2.1.2 Watts – Strogatz

The Watts-Strogatz graph model makes reference to the *k*-ring model explained in section 1.2.7. Consider a probability for rewiring the links in the ring, so that the average path length decreases very fast while the clustering coefficient still remains high. The rewiring can be considered as the process through which, with probability p, we replace each link of two nodes r-s with a link r-t, where t is a randomly chosen node different from r and s. If r-t is already contained in the modified network, no action is considered.

A variant of this process is to add links rather than rewire, which simplifies the analysis without considerably affecting the results.

2.1.3 Barabasi – Albert

The Barabasi–Albert model is based on two simple assumptions regarding network evolution:

- Growth: new nodes are added to the network, where each new node is connected to m existing nodes.
- Preferential attachment: this is the heart of the model. Each new node is connected to existing nodes with a probability proportional to its existing connections; the more connected, the more likely a node is to receive new links.

The network begins with an initial network of m_0 nodes ($m_0 \geq 2$), and the degree of each node in the initial network should be at least 1. New nodes are added to the network one by one. The probability, p, of a new node to be connected to an existing node i, is given by:

$$p(k)_i = \frac{k_i}{\sum_j k_j} \qquad \textbf{(2.2)}$$

Where j are all the pre-existing nodes.

2.1.4 Erdös – Rényi

The Erdös-Rényi graph model considers N nodes in an unconnected network distributed randomly. Therefore, all nodes connect each other by a certain probability p ($p > 0$), obtaining a set of pairs of nodes. Repeating this process M times, every node will have a degree M. If M is small in comparison with N, the network can be disconnected, with different sets of nodes disconnected one from another. In the other hand, if M is high, the network can hardly be disconnected.

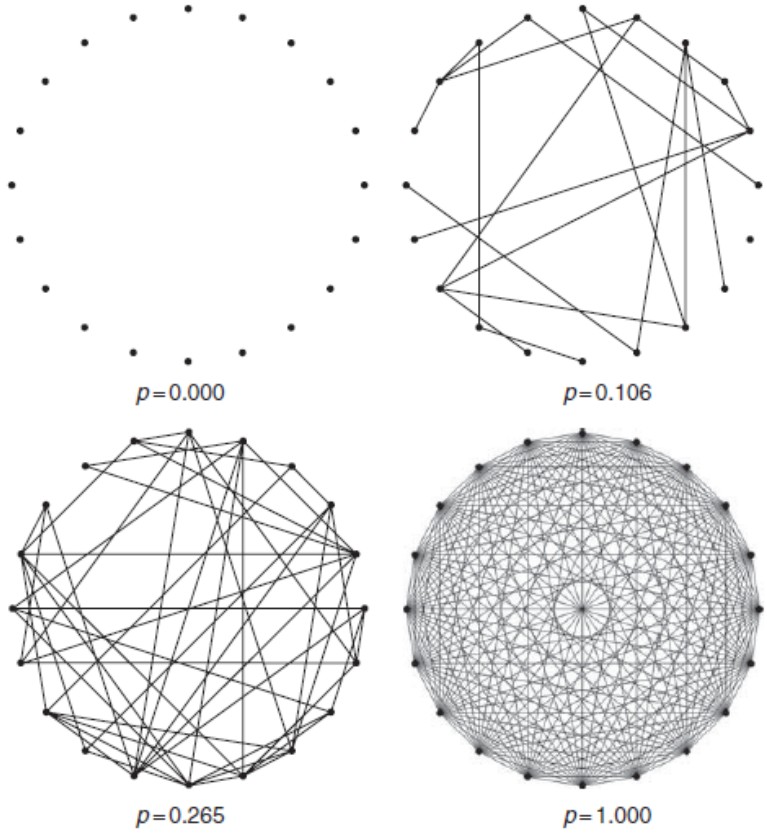


Figure 4. *Resulting graph depending on the value of the linking probability*

When nodes are linked this way, the degree distribution, $p(k)$, for a large number of nodes, follows a Poisson law.

$$p(k) = \binom{N-1}{k} P^k (1-P)^{N-1-k} \tag{2.3}$$

Where P is the probability of each node to be included in the graph.

2.1.5 Power-law Cluster

The Watts-Strogatz model shows a high clustering but without a scale-free degree distribution, while the Barabasi-Albert (BA) model with the scale-free nature does not has a high clustering. The power-law cluster model has *both,* a perfect power-law degree distribution and a high clustering.

It is essentially the BA growth model with an extra step to incorporate high clustering, after the preferential attachment (PA) step in the BA model algorithm we add an additional step called: *Triad formation*. If an edge between v and w was added in the previous PA step, then add one more edge from v to a randomly chosen neighbour of w.

This algorithm improves BA model in the sense that it enables a higher average clustering and more triangles.

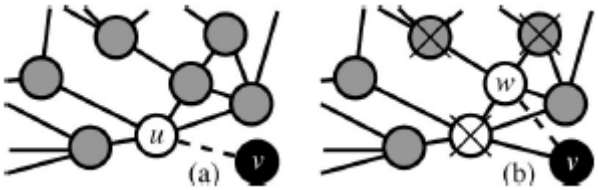


***Illustration 2**. Preferential attachment (a) and Trial formation (b)*

The standard scale-free network model suggests a mechanism for the emergence of power-law degree distributions in evolving networks: New actors (vertices) in a social context prefers to attach to more connected ('well known') actors. This mechanism of the emergence of clustering is well known, and was discussed under the name *'sibling bias'*.

In the power-law distribution, the probability of finding a node with degree k decays as a negative power of the degree: $p(k) \sim k^{-\gamma}$. This means that the probability of finding a high-degree node is relatively small in comparison with the high probability of finding low-degree nodes. Power-law relations are usually represented on a logarithmic scale, by which we obtain a straight line. [10]

2.1.6 Random Geometric

The random geometric graph model is the mathematically simplest spatial network. It is a model built by N nodes placed in a random position in a unit cube and linked by edges (e) according to a specified probability distribution:

$$p_e = \binom{N}{e} p^e (1-p)^{N-e} \quad \textbf{(2.4)}$$

Two nodes (u, v) are connected with an edge if and only if the distance between them is below a threshold.

$$d(u,v) \leq r \quad \textbf{(2.5)}$$

Where d is the Euclidean distance and r is a radius threshold.

This kind of graph resembles real human social networks in a number of ways. For instance, they spontaneously demonstrate community structure such as clusters of nodes with high modularity.

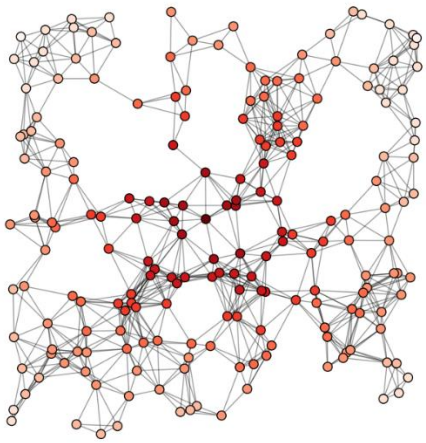


Illustration 3 *Example of a Random Geometric Graph*

2.1.7 Random Partition

The random partition graph model, also known as planted partition, represents community structures, where n vertices are classified into q groups. The groups have equal size and the vertices of the same group are linked with a probability p_{in}, while vertices of different groups are linked with a probability p_{out}. If $p_{in} > p_{out}$, the model graph has a built-in community structure. The vertex classification is indicated by the set of labels "{q_i}", which indicates a node inside a group.

The classification probability to distribute nodes in all groups is:

$$p(\{q_i\}) \propto \{\exp[-\sum_{(ij)} J\delta_{q_i q_j} - \sum_{i \neq j} J'\delta_{q_i q_j}/2]\}^{-1} \quad \textbf{(2.6)}$$

Where $J = \log\frac{p_{in}(1-p_{out})}{p_{out}(1-p_{in})}$, $J' = \log\frac{(1-p_{in})}{(1-p_{out})}$ and δ a density parameter of the groups.

2.1.8 Random Regular

Random regular graphs play a central role in combinatorics and theoretical computer science. It is a particular kind of random graph with *n* vertices, where $3 \leq d < n$ and $n \cdot d$ is even, but the regularity restriction modifies significantly the random properties. A random *d*-regular graph of large size is asymptotically almost surely d-connected, in other words, the probability of selecting a graph with a connectivity less than d tends to 0 as n increases. A d-regular graph has all vertices of degree d.

The pairing model is a method which takes $n \cdot d$ points and split them into n buckets with d points in each of them. Taking a random matching of the $n \cdot d$ points, and then linking the d points in each bucket into a single vertex,

yields a d-regular graph or multigraph. The resulting net should do not has multiple edges or loops, otherwise, we will need to start again. [16][17][18]

2.2.Centralities

The study of centrality addresses the question, “Which are the most important or central vertices in a network?”

In graph theory, we are particularly interested in the importance of a node due to its topological function in the network and possible applications in real life. To do so we are going to analyse each graph with several different centralities chosen to be the most common used.

2.2.1 Degree Centrality

It is the simplest measure in a network, is just the degree of a vertex, the higher the degree, the more a node is connected, and therefore, the higher is its centrality in the network. This centrality was considered by Wang and Rong [4] in the model of cascade-failure which motivates our study.

Although degree centrality is a simple centrality measure, it can be useful. In a social network, for instance, it seems reasonable to suppose that individuals who have connections to many others might have more influence, more access to information, or more prestige than those who have fewer connections.

2.2.2 Betweenness Centrality

Betweenness centrality counts all the paths going through a given node among all pairs of nodes in the network.

We can express the betweenness of a node i as:

$$x_i = \sum_{st} \frac{n^i_{st}}{g_{st}} \qquad \textbf{(2.7)}$$

Where s and t are the set of all pairs of nodes, g the number of geodetic paths between two nodes, in case that exists more than one, and $n^i{}_{st}$ the number of geodetic paths between s and t that go through i.

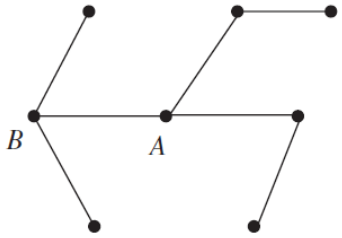


***Illustration 4**. Example of a simple graph. Node B has a higher Betweenness value than A*

The vertices with the highest betweenness are also the ones whose removal from the network will disrupt communications the most between other vertices. It measures how much a vertex connects others, so can occur that a vertex can have a low degree and high betweenness.

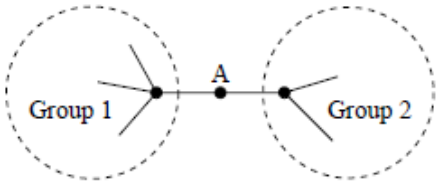


***Illustration 5.** Example of a node with a low degree and high betweenness.*

One application for the betweenness centrality is in identifying bottlenecks and important nodes in the network.

2.2.3 Communicability Centrality

The communicability centrality, also called subgraph centrality, is a method to characterize nodes in a network according to the number of closed walks starting and ending at the node. The contribution of these closed walks decreases as the length of the walks increases. In other words, shorter closed walks have more influence on the centrality of the vertex than longer closed walks.

Each closed walk is associated with a connected subgraph, this means that this measure counts the times that a node takes part in the different connected subgraphs of the network, having a higher importance in smaller subgraphs.

We define the communicability centrality of the vertex i as the number of closed walks of length k in the network starting and ending at vertex i. We can compute this value from the powers of the adjacency matrix A which give the number of walks between vertices.

But just counting the number of closed walks can take this number to infinite, we can avoid this problem by scaling the contribution of closed walks to the

centrality of the vertex by dividing them by the factorial of the order of the spectral moment ($k!$).

$$CommC(i) = \sum_{k=0}^{\infty} \frac{A_{ii}^{k}}{k!}$$

(2.8)

Communicability centrality takes into account not only the immediate effects of the closest nodes but also the long-range effects "transmitted" through the participation of a node in all subgraphs existing in the network. [15]

2.2.4 Closeness Current Flow Centrality

The closeness current flow centrality, also known as information centrality, is a variant of closeness centrality based on effective resistance between nodes in a network. It was introduced by Stephenson and Zelen in 1989 as a measure of node centrality of social networks. It is based on information that can be transmitted between any two points in a connected network.

Here a path connecting two nodes is considered as a "signal". The information measure (I_{ij}) between two nodes is defined as the reciprocal of the topological distance (d_{ij}) between the corresponding nodes.

$$I_{ij} = \frac{1}{d_{ij}}$$

(2.9)

Stephenson and Zelen proposed to define I_{ii} as infinite for computational purposes, which makes $1/I_{ii} = 0$.

The information centrality determines the average harmonic length of the paths that end in a node i:

$$InfoC(i) = \left[\frac{1}{n}\sum_{j}\frac{1}{I_{ij}}\right]^{-1}$$

(2.10)

This centrality will have a low value if a vertex is connected, through short paths, with a large number of nodes; it is better if a node has all nodes connected with the lowest number of steps possible, the centrality goes with the inverse of the average distance to other vertices. [15]

2.2.5 Pagerank Centrality

PageRank is the trade name given by Google, which uses it as a central part of their web ranking technology. The aim of the Google web search engine is to generate lists of useful web pages from a preassembled index of pages in

response to text queries. It does this by first searching the index for pages matching a given query using relatively simple criteria such as text matching, and then ranking the answers according to scores based on a combination of ingredients, of which PageRank is one.

Page Rank algorithm determines the importance of Internet pages based on the links pointing to them. There are three distinct factors that determine the PageRank of a node: the number of links pointing to the node, the number of links going outbound (link propensity) from the nodes that point you (linkers), and the centrality of your linkers. The nodes that have fewer links going outbound are worthier than the ones that have many. A node is important if it is highly linked, it is linked from other important nodes and linked by nodes with few outbound links. [22]

This algorithm initiates a random walk at a random node, following a random link at each node, with some small probability, at every step of jumping to a randomly chosen node without following a link. The algorithm gives high importance (high probability of hitting) to nodes with a high number of links pointing to them.

In mathematical terms, this centrality is defined by:

$$x_i = \alpha \sum_j A_{ij} \frac{x_j}{k_j^{out}} + \beta. \tag{2.11}$$

Where α is a normalization factor, k^{out} the number of links pointing out, A is a square matrix with rows and columns corresponding to web pages, and β a normalized source of a rank vector.

Note that the rank of a page is divided among its forward links to contribute to the ranks of the pages they point to, this covers both the case when a page has many backlinks and when a page has a few highly ranked backlinks. A page has a high rank if the sum of the ranks of its backlinks is high.

In our study, undirected graphs are converted to a directed graph with two directed edges for each undirected edge.

2.2.6 Eigenvector Centrality

The Eigenvector centrality computes the centrality for a node based on the centrality of its neighbours. The centrality value that is assigned to the nodes is based on the concept that connections to high-scoring nodes contribute more than equal connections to low-scoring nodes. A high eigenvector value means that a node is connected to many nodes who themselves have high scores.

To compute the centrality value let $A = (a_{v,t})$ be the adjacency matrix. If vertex v is linked to vertex t $a_{v,t} = 1$, and $a_{v,t} = 0$ otherwise. Furthermore, this can be generalized so that the entries in A can be real numbers representing connection strengths, as in a stochastic matrix.

The relative centrality value of a vertex v can be defined as:

$$x_v = \frac{1}{\lambda}\sum_{t \in G} a_{v,t} \cdot x_t \qquad \textbf{(2.12)}$$

Where λ is a constant. [24]

With a small rearrangement, this can be rewritten in vector notation as the eigenvector equation.

$$Ax = \lambda x \qquad \textbf{(2.13)}$$

In general, there will be many different eigenvalues λ for which a non-zero eigenvector solution exists. However, the additional requirement that all the entries in the eigenvector are non-negative implies that only the greatest eigenvalue results must be the corresponding eigenvectors. This the centrality out to be a revealing measure in many situations, for example, PageRank is a variant of Eigenvector centrality, and it is employed by the Google's Web search engine to rank Web pages.

The eigenvectors are only defined up to a common factor, so only centrality ratios of the vertices are well defined. To define an absolute score, it must be normalized in such a way that the sum over all vertices is 1 or the total number of vertices n.

3 Cascading Failure Simulations

3.1.Cascading failures in a network

A cascading failure in an interconnected system or network consists of successive node failures, triggered by the failure of a single node or of a subset of nodes. As the system must compensate for the failing components, other nodes become overloaded and also fail. Some models consider that the load of the failing nodes is distributed proportionally to their active neighbours according to their initial load, causing them too to collapse when they become overloaded. This process continues until either the load can be supported by the remaining nodes, or until the whole network fails.

Several years ago it was shown that real-world networks, such as those of transport or the Internet, are robust against random node failures but susceptible to intentional attacks. Although they start locally, depending on the structure of the network there are nodes that are particularly sensitive and a failure of any one of these could trigger a global network failure. For this reason, it is important to quickly and efficiently identify these weak links and secure them. One method would be to create a security system which allows the network to operate up to a pre-determined level of security, which would leave a margin big enough to absorb any possible extra load from neighbouring node failures. Working under these limits will ensure the proper functioning of the network. Otherwise, a prevention action protocol could be prepared in case of a failure to avoid a major catastrophe. [23]

3.2.Cascading failure simulation method

For this analysis, we will examine 200 different graphs belonging to 10 different categories. For each category created 20 different graphs are generated to allow a statistical study.

Each node starts with an initial load, L_j, which depends on the centrality that we are studying, and an initial maximum capacity, C_j.

$$L_j = [k_j(\sum_m k_m)]^\alpha \quad \textbf{(3.1)}$$

$$C_j = T \cdot L_j \quad \textbf{(3.2)}$$

Where k_j is the centrality of a node j, m runs on the set of neighbour nodes, α is a tunable parameter that controls the strength of the initial load and T is a tolerance parameter ($T \geq 1$). [4]

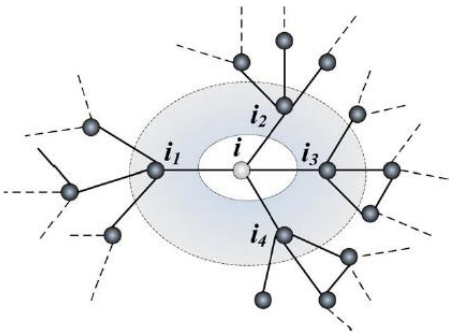


***Illustration 6**. An example of correlation between the initial load and its neighbours*

Once we have established these parameters, the next steps we will take are:

- We sort the nodes by their load L_j, and we select the 5 nodes with the highest load, the 5 with the lowest load and the 5 with a load value at the mid-point between the highest and the lowest.
  i.e.: if the highest value is 200 and the lowest 10, the nodes that we will take for the medium value analysis will be the 5 nodes closest to value 105.
- We will analyse of each individual graph and different load group separately.
- For every load group, we make these nodes (i) fail one by one and distribute their load to their active neighbours (j) as:

$$L_{j|new} = L_j + \Delta L_{ji} \quad \textbf{(3.3)}$$

$$\Delta L_{ji} = L_j \cdot \frac{L_j}{\sum_{m \epsilon \Gamma_i} L_m} \quad \textbf{(3.4)}$$

Where, ΔL_{ji} is the contribution of extra load received from the neighbours failed nodes and Γ_i the set of neighbours.

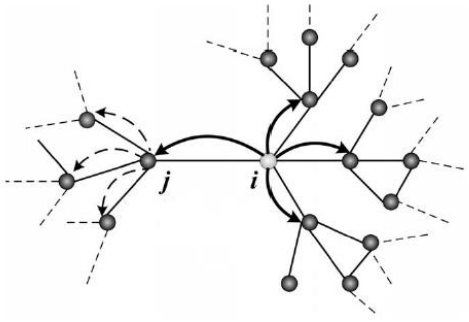


***Illustration 7** Load redistribution to neighbouring nodes connecting node i.*

A neighbouring node with a higher load will receive a higher shared load from the failing node.

- Once the load has been distributed to its neighbouring nodes, we check if any of the nodes affected has reached the C_j value. If so, it means that these nodes have failed and we have to redistribute this load. This procedure is repeated until the remaining active nodes stabilize or all the nodes fail.
- Once the cascade process is over, we count the nodes that have failed due to the node i failure, CF_i, and we repeat this process with the 5 nodes of the load group.
- To represent the robustness of the whole network as a number, compute the CF_{attack} as:

$$CF_{attack} = \frac{\Sigma_{i\epsilon A} CF_i}{N_A(N-1)} \quad \textbf{(3.5)}$$

Where A represents the set of nodes and N_A the number of nodes attacked.

When CF_{attack} =1 means that by failing 1 node, it collapses the whole network; CF_{attack} =0 means that by failing 1 node, none of the nodes is affected by it and their neighbours can absorb the extra load. The lowest the CF_{attack}, the more robust the network is.

3.3.Study Scenarios

With this study, we want to know which kind of nodes are more susceptible to intentional attacks. To do so we have created 10 different graph families, the most relevant, to analyse their behaviour:

- Geographical Threshold 2D
- Geographical Threshold 3D
- Watts – Strogatz (High Clustering)
- Watts – Strogatz (Low Clustering)
- Barabasi – Albert
- Erdös - Rényi
- Power-law Cluster
- Random Geometric
- Random Partition
- Random Regular

To create all these graphs we use the NetworkX [21] package of Python which allows us to easily create and manage graphs for the study. The code used to generate the graphs can be found in Annex B.

Furthermore, for each type of graph and to avoid errors of probability, we have randomly created 20 different graphs, each with 100 nodes and around 400 links.

Also, to ensure that our study takes into account the wide variety of networks in the real world, we have picked out the 6 most relevant centralities, and represented a wide range of topological node functions:

- Degree (Wang-Rong)
- Betweenness
- Communicability
- Closeness Current Flow
- Pagerank
- Eigenvector

Since α is a tunable parameter, we performed the analysis with 2 different values (α=0.1 and α=0.5) to compare their contribution to the process of cascade failure.

With all these variants we have a total of 2400 different graphs to study in our simulations. To simplify the analysis we examine 20 graphs belonging to each of the categories and take the average values. So, in the end, we just have 120 categories to analyse.

4 Results

4.1.Simulation Results

To run the simulations we have used the software Python 2.7 by using the Enthought Canopy program and the software package NetworkX 1.11

After running each simulation we obtained two file types which contained information about each scenario. One file gives information about the properties of the graph, shown in the following table, and how the networks reacted during the simulations.

Graph	***Diameter***	***AVG. Dist.***	***Clustering***	***Weiner Idx***
2D-Geo.	9,1	3,7758	0,6799	18690,3
3D-Geo.	6,1	2,8649	0,6549	14181,5
Barabasi-Albert	4,0	2,3691	0,1727	11727,2
Erdös-Rényi	4,2	2,4327	0,0832	12042,1
Hclust-Watts Strogatz	5,5	3,0766	0,4823	15229,4
Lclust-Watts Strogatz	4,0	2,4168	0,0726	11963,2
Powerlaw	4,0	2,3826	0,3317	11793,8
Random Geometric	11,1	4,5856	0,6387	22698,6
Random Partition	4,6	2,4626	0,0969	12190,0
Random Regular	4,0	2,4201	0,0642	11979,3

***Table 1**. Graph properties*

From *Table 1* we have selected some parameters about the graphs which describe and inform us about them. Take into account that these values are an average of the 20 graphs created per category.

If we examine the distance terms in this network, it shows that if we compare the diameter of a network to their average distance, we will find that this is very small; the diameter value is two to three times higher than the average distance. This phenomenon is due to the small-world effect.

In relation to the network's clustering coefficient, we can see that in half of the graphs (2D-Geo., 3D-Geo., Hclust-Watts-Strogatz, Powerlaw) these values show that close to half of the nodes share a connection to the other ones, which also contributes to the small-world phenomenon.

We have also plotted the degree histogram to show how the different graphs link their nodes. In *Figure 5* we have plotted a complete degree histogram in which we can observe at a first sight some relevant things.

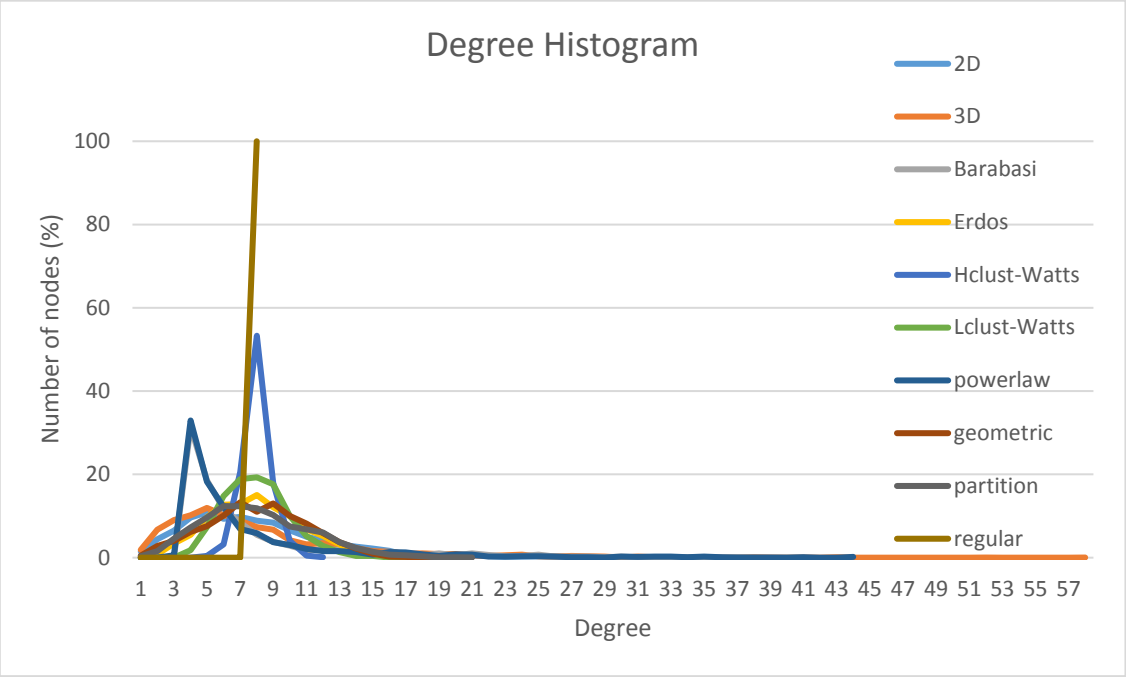


Figure 5. *Degree Histogram of all Graphs*

Figure 5 shows us is that the vast majority of the nodes have a low degree ranging between 1 and 15, compared to Random regular in which all nodes have the same degree (8), a degree between 1 and 15 means the network's nodes have an analogue distribution. On the other hand, there are some graphs that have nodes with a high degree, the highest is for 3D-Geographical Threshold, with a degree of 58, followed by Powerlaw, Barabasi-Albert and 2D-Geographical Threshold, but if you look at the histogram, then only around the 0.05-0.2% of nodes have these high degree values. This is why I have ignored this area and instead have focused on the zone where all nodes are concentrated, *Figure 6.*

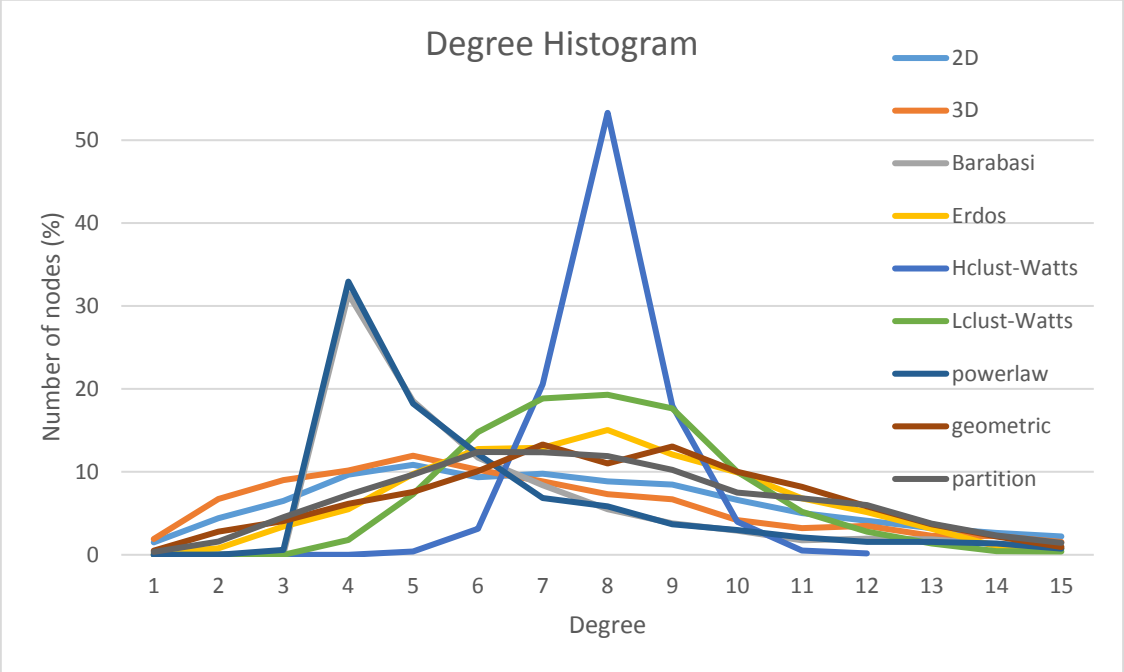


***Figure 6**. Zoom of the Degree Histogram*

In *Figure 6* we can observe that, in this interval, half of the graphs have a similar degree distribution. It appears that they follow a normal distribution with a standard deviation (σ^2) value of around 4 with a central axis between 7 and 8. The other half are quite different, especially *Watts-Strogatz Low-clustering*, *Barabasi-Albert* and *Powerlaw*, with these last two having an almost equal degree distribution. This similarity matches with the results in *Table 1*. This is because, as explained in chapter 2.1.5, the *Powerlaw* is like a *Barabasi-Albert* model which has been improved to achieve a high clustering.

This last example shows us that the clustering index doesn't have any relation to the degree histogram.

The second file obtained is a plot about CF_{attack} vs. Tolerance [*Annex C*]; based on publication [4], where the vulnerability of the USA power grid is tested using the algorithm explained in chapter 3.3.

Figures in *Annex C* show us when the graph starts to fail by overload due to an extra load from failed nodes, this let us know how robust they are from the *critical tolerance threshold* (T_c) parameter, which is the point in which the network begins to collapse, $CF_{attack} > 0$. This threshold can be seen as the minimum tolerance level to avoid a collapse. When $T > T_c$, it means that the network can work easily and can absorb more of the load. When $T = T_c$, it means that the system works well but care needs to be taken, as any inconvenience could reduce its capacity or increase the workload of any one node, causing it to collapse, and the load would then be transferred to the other nodes, in this way causing a cascade node failure. At this point, the network's tolerance is below T_c ($T < T_c$), as some nodes have failed and network's capacity is below the desired values. Thus, the higher the value of T_c, the more vulnerable the network is, therefore the network has a low robustness.

We have put these figures together in two different ways to compare them against one other and to help with the analysis. One classified by centrality, and the other by the graph family [*Annex C*]. Looking the simulation results we see some patterns emerge among the resulting figures.

Comparing centralities through a graph family, we have seen that:

- Medium Load (ML) Nodes always have more similarities to Low Load (LL) Nodes in Closeness Current Flow centrality than High Load (HL) Nodes as this does not occur with other centralities. This is because the centrality value in all nodes is not much different as it is in other centralities. Consequently, the values of ML nodes are close to those of LL nodes.
- We can also split the centralities into two different groups due to their similar behaviour at both "α" scenarios: Degree – PageRank – Closeness (this last centrality is a bit more different than the other two centralities, it has the above mentioned difference), and Eigenvector – Communicability – Betweenness. There are slight differences between both groups, but the most significant is their similar behaviour when changing from α=0.1 to α=0.5, and their T_c value. In the first group, the three node groups have a positive correlation between α and T_c. In other words, when α increases, T_c increases; in the second group, LL nodes have a negative correlation between α and T_c.
- The first centrality group has a notably higher T_c than the other, while Eigenvector, Communicability and Betweenness have the lowest T_c. Within each group, T_c has almost the same value, independent of the centrality.
- Comparing the centralities in Random Regular, we can see that all scenarios have exactly the same behaviour, there is no change. The three node groups display exactly the same behaviour among them and the other centralities, and even changing the "α" parameter there is no change. They are all completely equal.
- Analysing the resulting figures to know which centrality makes the nodes to fail the most, we have seen that HL and LL have completely different behaviour (ML follows HL behaviour), while Betweenness and Communicability take HL to their highest T_c values, they take LL to their lowest; and while Closeness and Pagerank take LL to their highest, they take HL to the lowest.

  However, when we look at which centralities give the network the highest incidence of failure, these are Closeness and Pagerank.

Comparing graphs through a centrality family, we have seen that:

- We can separate the graph families in different groups by their similar behaviour during the centrality families' comparison: 2D Geographical Threshold – 3D Geographical Threshold – Random Geometric, Barabasi-Albert – Powerlaw and Erdös-Rényi – Random Partition.

  In the first group, the graph completely fails at low tolerance values, but when we increase the tolerance compared to the others, the network

failure rate decreases gradually with small differences at each step. In the second group, the LL node keeps the network in total failure levels up to a high value of tolerance, after this point a small increase in tolerance causes a huge drop in failure levels. The third group is like the second one but with a gradual load absorption process, faster than the first group.

- HL nodes in the Communicability centrality for α=0.5, have the highest T_c values in comparison with the other centralities. For α=0.1, LL nodes have the highest tolerance values where the whole network is completely collapsed, what's more, HL nodes have the lowest T_c value.
- For α=0.5 in the Eigenvector centrality, in most of the graph types, the three node groups have exactly the same behaviour, you can superpose the lines one in front the other and there is no deviation. There are only three types that differ from the others and this is only by the LL nodes: 2D-Geographical threshold, 3D-Geographical Threshold and Random Geometric. The one that differs the most is Random Geometric, 3D-Geographical differs the least.
- Analysing the result figures to know which graph family makes their nodes fail the most, we have seen some differences among centralities. For HL, the graph family that takes the network to higher T_c values are Watts-Strogatz High-clustering or Random Regular depending on the centrality, Barabasi-Albert and Power-law take the network to lower T_c values. For LL, the graph family that takes the network to higher T_c values is Random Geometric; Random Regular to lower T_c values. Thus, we can observe that the graph family that keeps the network's nodes in high or low failure values depends on the centrality.

Furthermore, in both comparisons we have noticed that there are some common behaviours:

- In most cases, ML Nodes always have either none or just a few changes in behaviour between α=0.1 and 0.5, with a T_c value between 1.10 and 1.15.
- For α=0.5, the three node groups have more similar behaviour between one another than compared to α=0.1 for every scenario. They converge for ML behaviour, HL nodes have almost the same behaviour as ML, however LL nodes, in the majority of cases, is completely different. LL nodes can be seen as the group that changes their behaviour the most, having the lowest T_c, especially for the betweenness and communicability centralities.
- LL nodes are the group that fails the most, as the graph needs more tolerance to resist a failure. On average, if LL nodes fail, the graph completely fails, even at a tolerance between 15-20% ($CF_{attack} = 1$), and in the worst cases at values of 30%. However, LL nodes produce a gradual failure instead of going from $CF_{attack} = 0$ to $CF_{attack} = 1$ suddenly or in 2 or 3 steps, like ML and HL nodes do.

4.2.Real-life Scenario Results

Taking into account the results obtained in the previous chapter, we are going to test them in a real-life scenario: the USA air transport network in 2001.

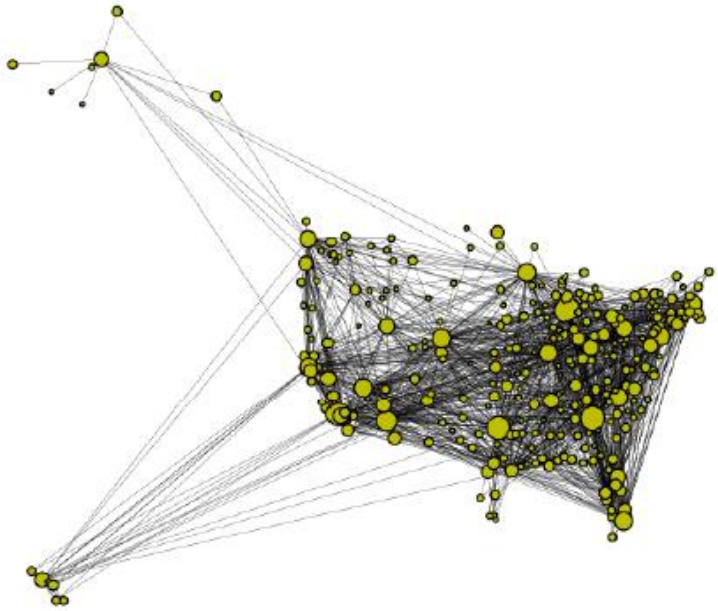


Illustration 8. *Representation of the USA air transport network in 2001.*

Firstly, it is important to know some information about the network:

Graph	***Nodes***	***Edges***	***Diameter***	***AVG. Dist.***	***Clustering***	***Weiner Idx***
USA	295	2072	5	2.4594	0.7046	106656.0

Table 2. *USA-graph properties*

If we have a look at *Table 2* and compare it with the properties obtained by the graph we have created and analysed in the previous chapter, we can see that the graph with the most similar properties is: 3D Geographical Threshold. This would not be surprising, the USA-graph is a network of aircraft routes through the USA territory, with aircraft flying over airports without stopping there, linking distant nodes directly; therefore it is a good example to imagine a 3D Geographical Threshold graph.

But having a graph with similar properties doesn't necessarily mean that their behaviour will be similar, we need to compare their behaviour explicitly and see which ones have more similar behaviour. To do so, we are going to take the USA-graph, plot the different behaviours of every different centrality [Annex D.1] and compare them against the figures obtained for the previous chapter.

Having a quick look through the figures obtained [Annex D], it can be seen that only some of them have similar behaviour, in the majority of centralities, the three node groups follow the same patterns and their T_c values are similar

(especially for α=0.1, at α=0.5 USA-graph doesn't give much information, mostly graphs suddenly recover completely at the second or third step). These graphs are:

- 2D-Geographical Threshold
- 3D- Geographical Threshold
- Random Geometric
- Barabasi-Albert
- Erdös-Renyi

In chapter 4.1, we comment that 2D and Random Geometric are similar to 3D, so it could be reasonable that they are similar to the USA air transport network.

Then, the graphs must be compared in pairs against the USA air transport network to know which ones have the highest similarity. To do so we have created the following table to compare the graphs and identify the most similar graph family.

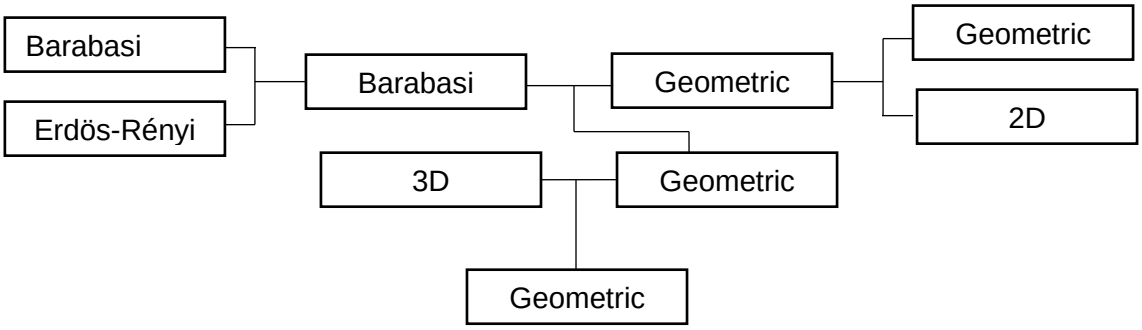


Illustration 9. *Comparison groups against USA-graph.*

(It is necessary to comment that USA-graph is not similar to any of the theoretical behaviours, so it was hard to conclude which one was most similar, but there were some patterns and parameters that make one think that it actually follows one of the structures studied)

Surprisingly, the graph family that matches better with the real network behaviour is Random Geometric, rather than 3D Geographical Threshold as was expected.

This can be explained by the hub structure of the USA's airlines. Each airline bases its operational model on having some hubs around the country plus some "mini-hubs". These "mini-hubs" help airlines to cover all the secondary airports.

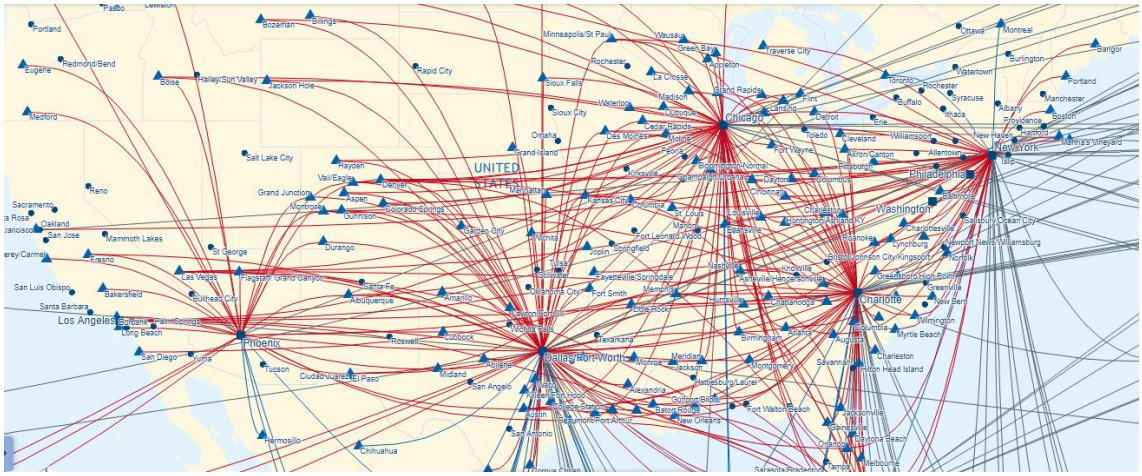


Illustration 10. *Destinations network of American Airlines from NY (Jan. 2018)*

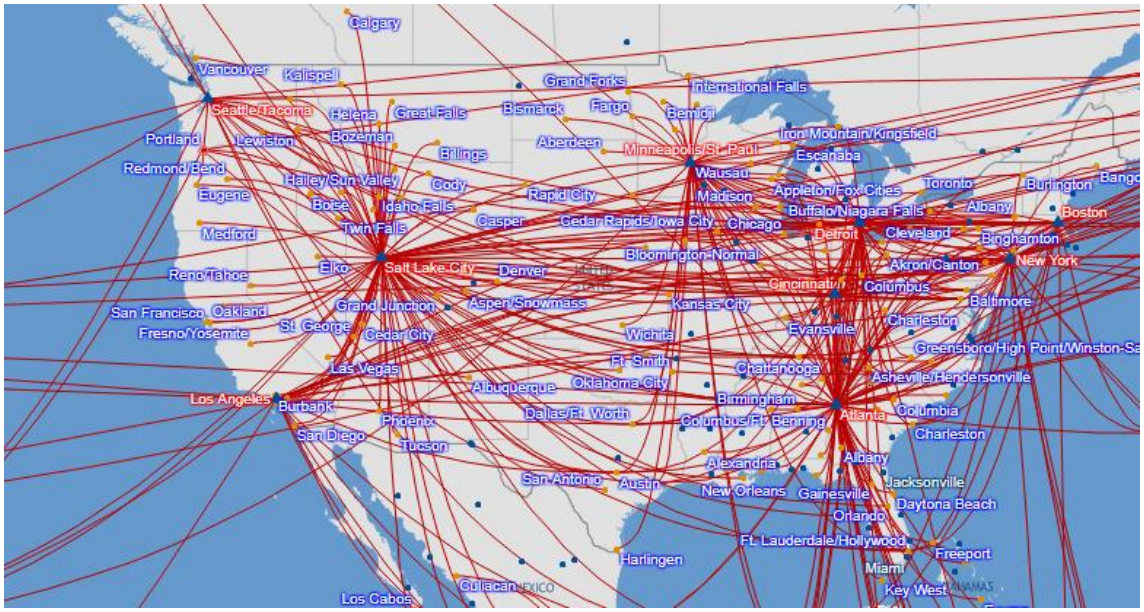


Illustration 11. *Destinations network of Delta Airlines from Salt Lake City (Jan. 2018)*

This kind of structure creates a modular network where, inside each group, small airports are only connected to a hub and hubs connect to other groups as it can be seen in *Illustration 8, 10 and 11.*

That is why Random Geometric is the graph family that fits best with the simulation behaviour results. To answer why the properties match with 3D Geographical Threshold could be explained by the fact that in the USA-graph there are 3 times more nodes and 5 times more edges. This can mean that structurally, the network looks like a 3D, but the ratio between nodes and edges is different as there are more edges, and therefore the network is much more connected. Thus, the diameter and the average distance have been reduced, making the network much more similar to 3D than Geometric, but it still has an intrinsic Geometric structure, as these new edges have not modified the network.

Another comparison we have done with USA-graph figures is about which centrality makes the network fail the most by comparing each node group against the centralities [Annex D.2] to know which nodes are more likely to fail.

Looking at the figures, it makes no doubt that the centrality that fails the most is Current Flow Closeness, followed by Pagerank. The one that fails the least is Betweenness, except for HL, which is the one which fails the most, but if we have a look at the created graphs, the centralities that fail the least are Communicability and Eigenvector followed by Betweenness.

5 Conclusions

Our lives involve many complex networks dealing with different aspects of our daily routines such as telecommunications, power grid, sewage system, the internet or transport networks for goods or passengers. It is important to know and understand how these networks evolve and which parameters can rapidly modify their behaviour, in order to be able to develop high-tolerance networks which are less affected against intentional attacks or random failures.

For the purpose of this study, we have considered different families of graphs and different centrality measures to study their failures from five initial nodes. By analysing and comparing the results obtained, we found that when considering a given graph family, the failure process is similar among different centralities and does not depend on the initial node. However, if we compare the results considering their centrality measure, the results completely differ among them. Thus, we can state that the behaviour of a network failure mainly depends on the type of graph rather than on the centrality considered.

In addition, there exists a tunable parameter that controls the strength of the initial load which we call “α”. A higher “α” means that the network is less prone to failure. When we compare results between α=0.1 and α=0.5, the node groups HL (High Load) and LL (Low Load) have different behaviours for every graph family and centrality. This happens with LL and ML (Medium Load), not with HL nodes which increase their critical tolerance (T_c) values, the point in which the network starts to collapse. Therefore, these findings suggest that for the HL node groups it is preferred to analyse networks with a high “α” value, however, it is more relevant to focus on LL node groups due to their high importance in cascade failures.

Besides, we consider relevant to know which parameters affect the most to a network failure for two main reasons. First, results among graph families are difficult to compare, as each of them has completely different patterns, and there are only a few similar parameters among them. Second, results among different centralities follow a much clear pattern in all figures. Centralities that make the network more robust against failures are Betweenness and Communicability, as these centralities base their initial load according to the nodes that are few links away to them, no matter how big the network is and how the distant nodes are connected. A high load means that the nodes have a well-connected cluster around them. The centralities that make the network more vulnerable against intentional failures, even at high tolerance values, are Closeness Current Flow and Pagerank. These centralities base their initial load according to how the whole network is connected and, if the nodes are well-connected to the rest of nodes in the network, preferably through the fewest possible number of steps. For this, the graphs families which have more failing nodes at these centralities, are the ones with the highest diameter and average distance between a pair of nodes: 2D/3D-Geographical Threshold and Random Geometric. Hence, to increase network’s resistance, we need to look at each node and analyse how it is connected to the nodes around them.

Escola d'Enginyeria de Telecomunicació i Aeroespacial de Castelldefels
UNIVERSITAT POLITÈCNICA DE CATALUNYA

ANNEXOS

TÍTOL DEL TFG: Cascade failures in complex networks

TITULACIÓ: Grau en Enginyeria d'Aeronavegació

AUTOR: Llorenç Sánchez Marín

DIRECTOR: Francesc Comellas Padró

DATA: 30 de Maig de 2018

6 Annexes

6.1. Annex A – Code

```
import time
import itertools
import statistics
import networkx as nx
from pylab import *

starttime = time.time()

####################################
### Starting funciton. Compute initial load Lj
####################################
def ini_carr(alpha,tol):
for j in g:
kj=gload[j]
v=g.neighbors(j)
sumkm=0
for m in v:
sumkm=sumkm+gload[m]
Lj=(kj*sumkm)**alpha
g.node[j]['Lj']=round(Lj,4)
carr_max(tol)

####################################
### Compute maximum load Cj for every node
####################################
def carr_max(T):
for j in g:
Cj=(T*g.node[j]['Lj'])
g.node[j]['Cj']=round(Cj,4)

####################################
### Distribution of node's load to live neighbours
####################################
def distrib_carr(i):
sum_carr_veins=0
for j in g.neighbors(i):
if g.node[j]['OK']==1:
sum_carr_veins=sum_carr_veins+g.node[j]['Lj']  #si tots els veïns OK=0?
for j in g.neighbors(i):
if g.node[j]['OK']==1 and sum_carr_veins!=0:
newLj=g.node[j]['Lj']+g.node[j]['Lj']*g.node[i]['Lj']/sum_carr_veins # si Lj=0;
newLj=Lj(antiguo)
g.node[j]['Lj']=round(newLj,4)

####################################
```

```
### Sort nodes in a list from min to max Cj
####################################

def listdownup():
gsort=sorted(g.nodes_iter(data=True), key=lambda labels: labels[1]['Cj'])
## gsort=sorted(g.nodes_iter(data=True), key=lambda labels:
labels[1]['Cj'],reverse=True)
ordlist=[]
for i in range(len(gsort)):
nod=gsort[i][0]
ordlist.append(nod)
return ordlist


#### Per invertir la llista downup
## updown=downup.reverse()

####################################
#Funcio cascada a partir d'un node fa la cascada i retorna la llista dels que han falla la
llista es una llista de llistes a cada posicio diu els nodes que han fallat al pas 0,1,2,etc
####################################

def cascada(v):
eccg=nx.eccentricity(g)
ecc=eccg[v]
for i in g:
g.node[i]['OK']=1
g.node[v]['OK']=0
dist=-1
finit=0
oldlevel=[v]
hanfallat=[oldlevel]
while (finit==0):
dist=dist+1
newlevel=[]
for pos in oldlevel:
distrib_carr(pos) #new neighbour's node load
#print (v,pos)
for j in g.neighbors(pos): #which nodes are failing?
if g.node[j]['OK']==1:
if ((g.node[j]['Lj']) >= (g.node[j]['Cj'])):
g.node[j]['OK']=0
newlevel.append(j)
hanfallat.append(newlevel)
oldlevel=newlevel
if (dist==ecc):
finit=1
return hanfallat
###############
def FCcommunicability_exp(G):
```

```
import scipy.linalg
nodlst = list(G) # ordering of nodes in matrix
A = nx.to_numpy_matrix(G,nodlst)
# convert to 0-1 matrix
# convert to 0-1 matrix
A[A!=0.0] = 1
# communicability matrix
expA = scipy.linalg.expm(A.A)
mapping = dict(zip(nodlst,range(len(nodlst))))
c = {}
for u in G:
c[u]={}
for v in G:
c[u][v] = float(expA[mapping[u],mapping[v]])
return c
# ########################################
# Communicability distance index (big Gamma)

def comm_dist_idx(G):
Gamma_idx=0.0
cdst=FCcommunicability_exp(G)
for u in G:
for v in G:
val=cdst[u][u]+cdst[v][v]-2*cdst[u][v]
val=abs(val)
## if val<0:
## print "problemo: ",val,"u: ",u,"v: ",v,"uu: ",cdst[u][u],"vv: ",cdst[v][v],"uv:
",cdst[u][v]
## val=-val
Gamma_idx+=sqrt(val)
return Gamma_idx/2

def wiener_idx(G, weight=None):
# compute sum of distances between all node pairs
# (with optional weights)
wiener=0.0
if weight is None:
for n in G:
path_length=nx.single_source_shortest_path_length(G,n)
wiener+=sum(path_length.values())
else:
for n in G:
path_length=nx.single_source_dijkstra_path_length(G,n,weight=weight)
wiener+=sum(path_length.values())
return wiener/2.0
def Q_idx(G,wieneridx):
return log(0.857763885*wieneridx/comm_dist_idx(G))
```

```
def correlation(xlist, ylist):
xbar = mean(xlist)
ybar = mean(ylist)
xstd = standardDev(xlist)
ystd = standardDev(ylist)
num = 0.0
for i in range(len(xlist)):
num = num + (xlist[i]-xbar) * (ylist[i]-ybar)
corr = num / ((len(xlist)-1) * xstd * ystd)
return corr
def standardDev(alist):
theMean = mean(alist)
sum = 0
for item in alist:
difference = item - theMean
diffsq = difference ** 2
sum = sum + diffsq
sdev = math.sqrt(sum/(len(alist)-1))
return sdev

####################################
### Part principal
####################################

CFlist20LL=[]
CFlist20HL=[]
CFlist20ML=[]

#filename = "geographical_threshold/dim2/geographical_threshold_2D_100--"
#grafname = "2D-geographical_threshold-"

#filename = "geographical_threshold/dim3/geographical_threshold_3D_100--"
#grafname = "3D-geographical_threshold-"

#filename = "watts_strogatz/H/wattsst_Hclust_100--"
#grafname = "Hclust-watts_strogatz-"

filename = "watts_strogatz/L/wattsst_Lclust_100--"
grafname = "Lclust-watts_strogatz-"

#filename = "barabasi_albert/barabasi_albert100--"
#grafname = "Barabasi_albert-"

#filename = "erdos_renyi/erdos_renyi100--"
#grafname = "Erdos_renyi-"

#filename = "powerlaw_cluster/powerlaw_cluster100--"
#grafname = "Powerlaw_cluster-"

#filename = "random_geometric/random_geometric_100--"
```

```
#grafname = "Random_geometric-"

#filename = "random_partition/random_partition100--"
#grafname = "Random_partition-"

#filename = "random_regular/random_regular_100--"
#grafname = "Random_regular-"

centrality= 'Eigenvector '

reps=20
alfa=0.5
alpha='0_5'
dt = 0.02
ngraus=5
tolvals = arange(1.00, 1.50, dt)
granslist=[]
petitslist=[]
miglist=[]
gransnodelist=[]
petitsnodelist=[]
mignodelist=[]

corr=[]
diameter=[]
avgdist=[]
clust=[]
wiener=[]
qidx=[]
histgrm=[]

hstgrmlenmax=0

for idx in range(reps):
#idx=9

g=nx.read_edgelist(filename+str(idx)+".edges",nodetype=int)
gload=nx.eigenvector_centrality_numpy(g, weight='weight')
diameter.append(nx.diameter(g))
avgdist.append(nx.average_shortest_path_length(g))
clust.append(nx.average_clustering(g))
wiener.append(wiener_idx(g))
qidx.append(Q_idx(g,wiener[idx]))
hstgrm=nx.degree_histogram(g)
histgrm.append(hstgrm)
if len(hstgrm)>hstgrmlenmax:
hstgrmlenmax=len(hstgrm)
Gedges=g.edges()
xlist=[]
```

```
ylist=[]
for i in range(0,g.size()):
xlist.append(g.degree(Gedges[i][0]))
xlist.append(g.degree(Gedges[i][1]))
ylist.append(g.degree(Gedges[i][1]))
ylist.append(g.degree(Gedges[i][0]))
corr.append(correlation(xlist, ylist))

print time.asctime( time.localtime(time.time()) )
print 'file:',grafname+str(idx)+'.edges'
print 'avg clus-coeff.= %-6.4f ' % (nx.average_clustering(g))
print 'alpha:',alfa
print nx.info(g)
print
###############
CFlist=[]
tol=1.00
ini_carr(alfa,tol) #Asigna una carga a cada nodo (Todos activos)
while (tol<1.50):

if len(CFlist)==0:
load=listdownup() #sort nodes, saves nodes names
load.reverse()
## print graus
grans=[load[i] for i in range(ngraus)]
#print '\nGRANS '+str(grans)
gransload=[]
for i in grans:
gransload.append(round(gload[i],4))
print '\nGRANS '+str(grans)
print ' alfa:',alfa,' T: ',tol
start = time.time()
faillist=[]
#f.write('\n GRANS \nTolerance: '+str(tol))
for i in grans:
ini_carr(alfa,tol)
fail=cascada(i) #nodes that failed
flat=list(itertools.chain.from_iterable(fail)) #give all nodes in 'fail'
#print flat
#f.write('\n'+str(flat))
CFi= len(flat)
faillist.append(CFi)
elapsed = (time.time() - start)
print 'Nodes failing: '+str(faillist)
CFattck=round(sum(faillist)/(ngraus*(g.order()-1)*1.0),4)
print 'CFattck: ',CFattck
CFlist.append(CFattck)
#print time.asctime( time.localtime(time.time()) )
#print elapsed
tol=tol+dt
```

```
CFlist20HL.append(CFlist)
gransnodelist.append(grans)
Lj0=[]
Ljn=0
for i in grans:
Ljn=g.node[i]['Lj']
Lj0.append(Ljn)
inf=zip(grans, gransload,Lj0)
granslist.append(inf)

############

CFlist=[]
tol=1.00
ini_carr(alfa,tol)
while (tol<1.50):
if len(CFlist)==0:
load=listdownup()
#graus.reverse()
## print graus
petits=[load[i] for i in range(ngraus)]
petitsload=[]
for i in petits:
petitsload.append(round(gload[i],4))
print '\nPETITS '+str(petits)
print ' alfa:',alfa,' T: ',tol
start = time.time()
faillist=[]
#f.write('\n PETITS \nTolerance: '+str(tol))
for i in petits:
ini_carr(alfa,tol)
fail=cascada(i)
flat=list(itertools.chain.from_iterable(fail))
#print flat
#f.write('\n'+str(flat))
CFi= len(flat)
#if CFi==1 and ftol[petits.index(i)]==0: #get node tolerance
# ftol[petits.index(i)]=round(tol,4)
faillist.append(CFi)
elapsed = (time.time() - start)
print 'Nodes failing: '+str(faillist)
CFattck=round(sum(faillist)/(ngraus*(g.order()-1)*1.0),4)
print 'CFattck: ', CFattck
CFlist.append(CFattck)
#print time.asctime( time.localtime(time.time()) )
#print elapsed
tol=tol+dt
CFlist20LL.append(CFlist)
petitsnodelist.append(petits)
```

```
Lj0=[]
Ljn=0
for i in petits:
Ljn=g.node[i]['Lj']
Lj0.append(Ljn)

inf=zip(petits, petitsload, Lj0)

petitslist.append(inf)

###############
CFlist=[]
tol=1.00
ini_carr(alfa,tol)
while (tol<1.50):
if len(CFlist)==0:
load=listdownup()
load.reverse() #from highest to lowest
grausload=[]
for i in load:
grausload.append(gload[i])
## print graus
#hstgrm=nx.degree_histogram(g)
load_max=gload[load[0]]
load_mig=(load_max)/2
for i in range(len(load)):
ns=grausload[i]-load_mig
if ns<0:
if abs(grausload[i]-load_mig)-abs(grausload[i-1]-load_mig)>0:
i=i-1 #We choose the closest value to load_mig
break
#range of mitjans
mitjans=[]
j=i
if i!=len(load)-1: #if i=99; middle node is the lowest node
mitjans.append(load[i])
else:
mitjans=petits
while len(mitjans)<ngraus:
ns=abs(grausload[i-1]-load_mig)-abs(grausload[j+1]-load_mig)
if ns<0 and i>0: #we choose the closest values to load_mig
mitjans.append(load[i-1])
i=i-1
#if i==0:
# mitjans=grans
elif ns>=0 and j<(len(load)-1):
mitjans.append(load[j+1])
j=j+1
if j==99:
mitjans=petits
```

```
mitjansload=[]
for i in mitjans:
mitjansload.append(round(gload[i],4))
print '\nMITJANS '+str(mitjans)
print ' alfa:',alfa,' T: ',tol
start = time.time()
faillist=[]
#f.write('\n MITJANS \nTolerance: '+str(tol))
for i in mitjans:
ini_carr(alfa,tol)
fail=cascada(i)
flat=list(itertools.chain.from_iterable(fail))
#print flat
#f.write('\n'+str(flat))
CFi= len(flat)
faillist.append(CFi)
elapsed = (time.time() - start)
print 'Nodes failing: '+str(faillist)
CFattck=round(sum(faillist)/(ngraus*(g.order()-1)*1.0),4)
print 'CFattck: ',CFattck
CFlist.append(CFattck)
tol=tol+dt
CFlist20ML.append(CFlist)
mignodelist.append(mitjans)
Lj0=[]
Ljn=0
for i in mitjans:
Ljn=g.node[i]['Lj']
Lj0.append(Ljn)
inf=zip(mitjans, mitjansload, Lj0)

miglist.append(inf)
#f.close()

###############
AVGdiameter=statistics.mean(diameter)
AVGavgdist=round(statistics.mean(avgdist),4)
AVGclust=round(statistics.mean(clust),4)
AVGwiener=statistics.mean(wiener)
AVGqidx=round(statistics.mean(qidx),4)
AVGcorr=round(statistics.mean(corr),4) # random regular AVGcorr=1
AVGhistgrm=[]

for i in range(hstgrmlenmax):
hist=[]
for j in range(len(histgrm)):
if len(histgrm[j])>i:
hist.append(histgrm[j][i])
else:
```

```
hist.append(0)
AVGhistgrm.append(statistics.mean(hist))

###############
print

dt = 0.02
t = arange(1.00, 1.50, dt)

processtime = time.time()-starttime

f=open(grafname+centrality+'-alpha'+alpha+'--failure_info.txt','w')
f.write('\ngraf type: '+grafname)
f.write('\ncentrality: '+centrality+'\n')
f.write('\nreps: '+str(reps)+'\nalpha:'+str(alfa)+'\ngraus:'+str(ngraus)+'\n')
f.write('\nprocess time [s]: '+str(processtime)+'\n')
#f.write('\n gransini: '+str(granslist)+'\n petitssini: '+str(petitslist)+'\n')
f.write('\ntolvals: '+str(tolvals)+'\n')
f.write('\n diameter: '+str(AVGdiameter)+'\n avg distance:'+str(AVGavgdist)+'\n
clustering:'+str(AVGclust)+'\n Wiener index:'+str(AVGwiener)+'\n Q
index:'+str(AVGqidx)+'\n Correlation:'+str(AVGcorr)+'\n')
f.write('\n avg histogram: '+str(AVGhistgrm)+'\n')
f.write('\n\nHL\n[node '+centrality+'
initial_load]\n'+str(granslist)+'\n\n'+str(CFlist20HL)+'\n')
f.write('\n\nLL\n[node '+centrality+'
initial_load]\n'+str(petitslist)+'\n\n'+str(CFlist20LL)+'\n')
f.write('\n\nML\n[node '+centrality+'
initial_load]\n'+str(miglist)+'\n\n'+str(CFlist20ML)+'\n')
f.close()


print tolvals
hl = [round(float(sum(col))/len(col),4) for col in zip(*CFlist20HL)]
ll = [round(float(sum(col))/len(col),4) for col in zip(*CFlist20LL)]
ml = [round(float(sum(col))/len(col),4) for col in zip(*CFlist20ML)]
plot(tolvals, hl,'r-',marker='p')
plot(tolvals, ll,'b-',marker='o')
plot(tolvals, ml,'g-',marker='x')
axis([1,1.3,0,1.1])
tcks = arange(1.00, 1.50, 0.05)
xticks(tcks)
xlabel('T')
ylabel('CFattack')
title(grafname+'-'+centrality+' alph='+str(alfa)+' red:HL blue:LL green:ML
rep'+str(reps))

##
savefig(grafname+'-'+centrality+'-alph'+alpha+'.png')

show()
```

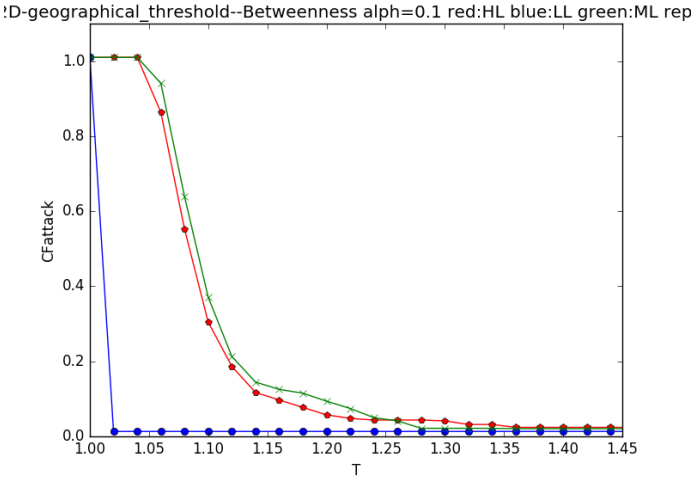
6.2. Annex B – Graph creation functions

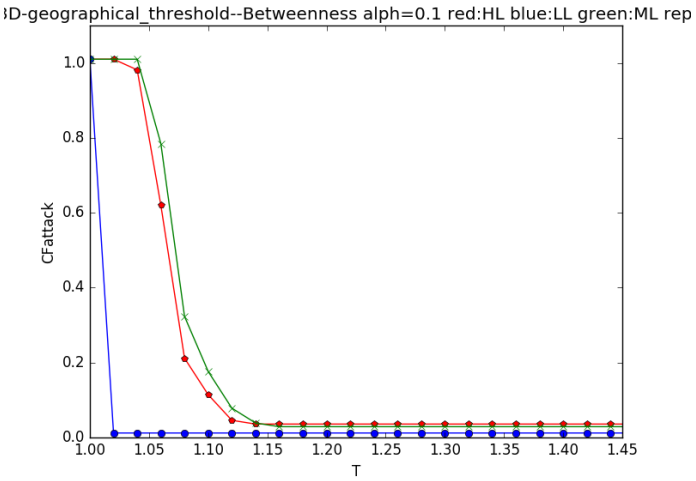
```
graf=nx.barabasi_albert_graph(100,4)
graf=nx.erdos_renyi_graph(100, 0.08)
graf = nx.geographical_threshold_graph(100,63,dim=2)
graf=nx.geographical_threshold_graph(100,21,dim=3)
graf=nx.powerlaw_cluster_graph(100,4,0.5)
graf=nx.random_geometric_graph(100,0.175)
graf = nx.random_partition_graph([5,40,20,10,15,10],0.19,0.05)
graf= nx.random_regular_graph(8, 100)
graf=nx.connected_watts_strogatz_graph(100, 8, 0.1) #high clustering
graf= nx.connected_watts_strogatz_graph(100, 8, 0.8) #low clustering
```

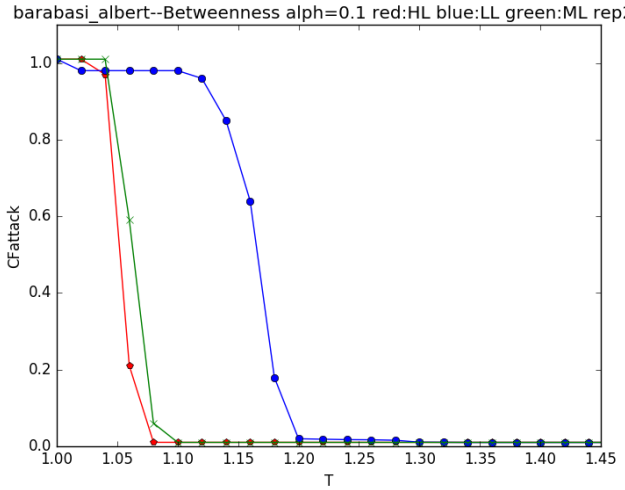
6.3. Annex C – Graph plots

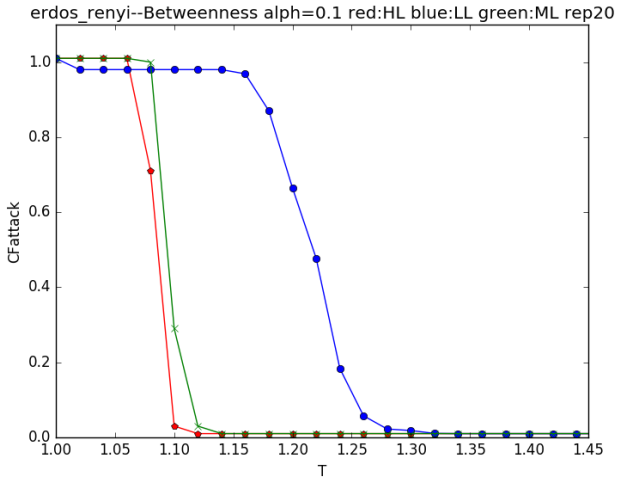
6.3.1 Sorted by Centrality family

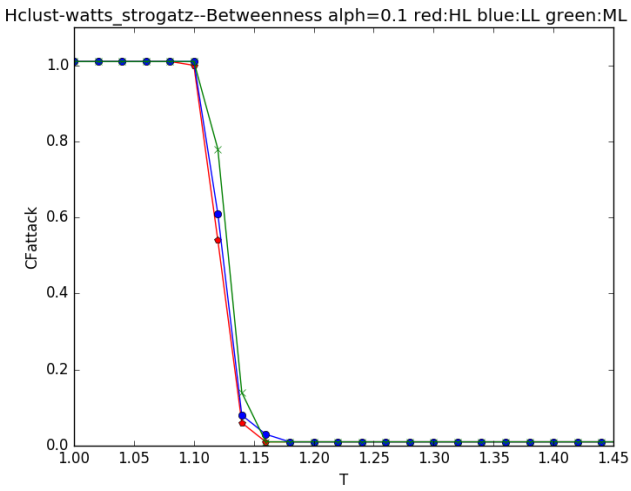
Betweenness 0.1

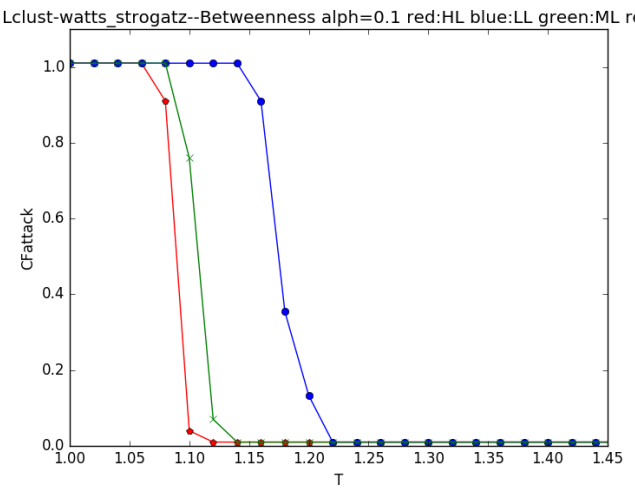


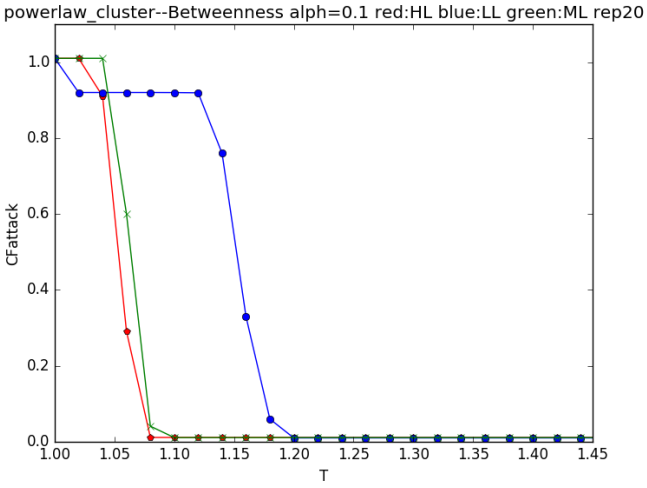


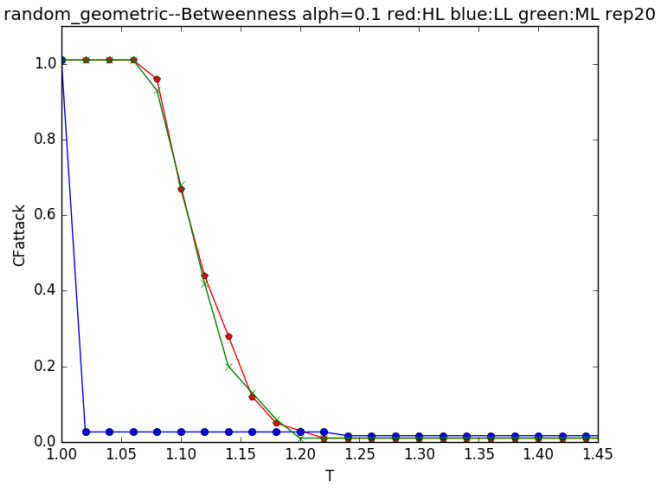


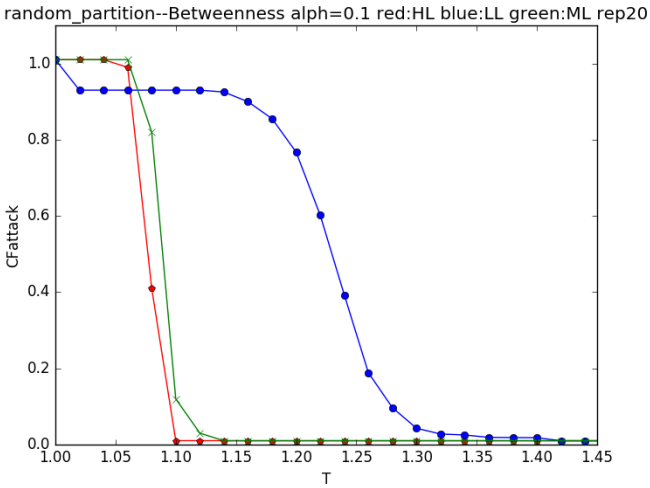


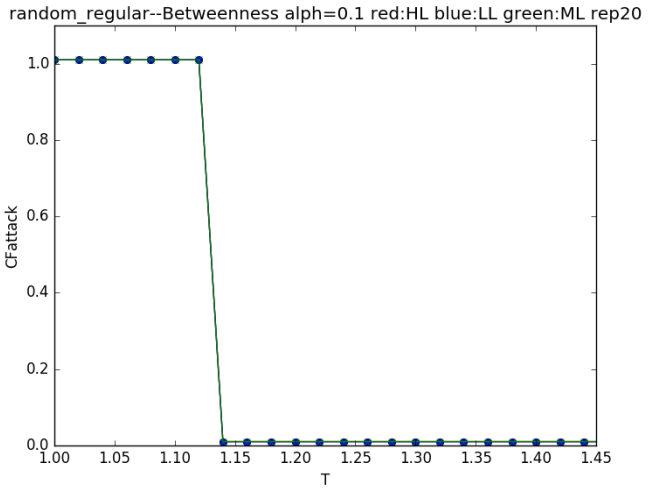




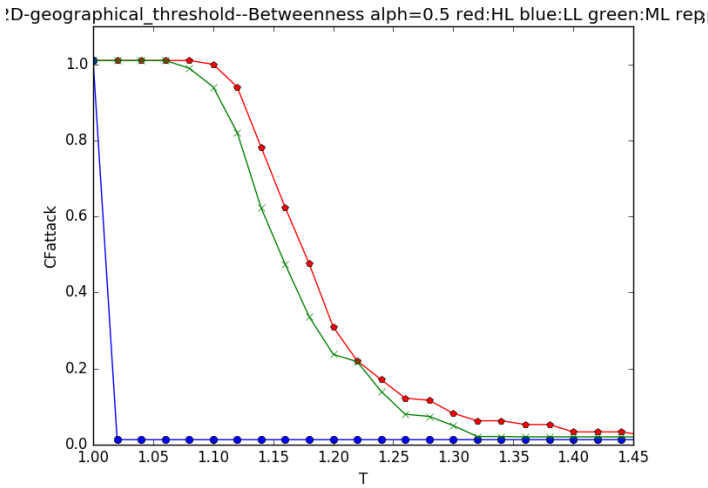


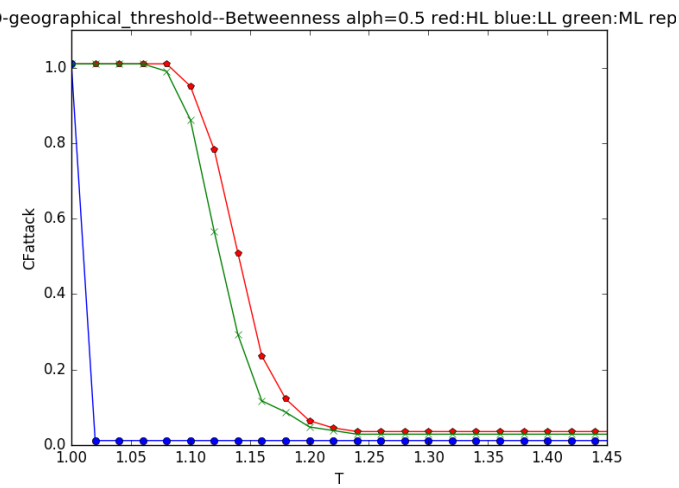






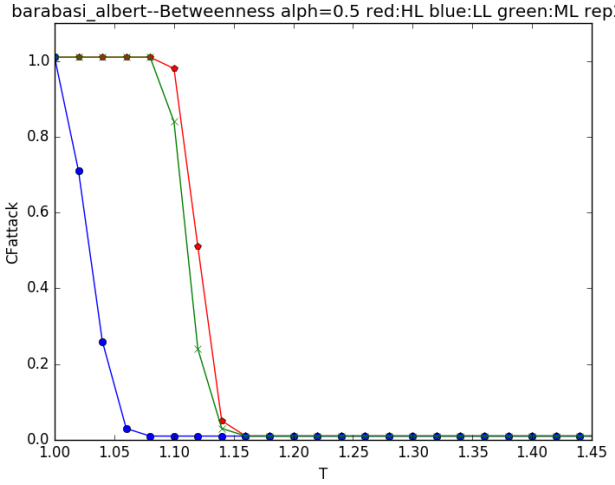
Betweenness 0.5



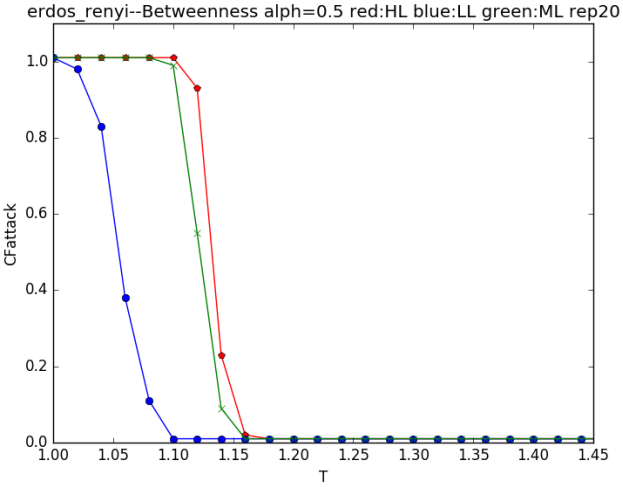


---

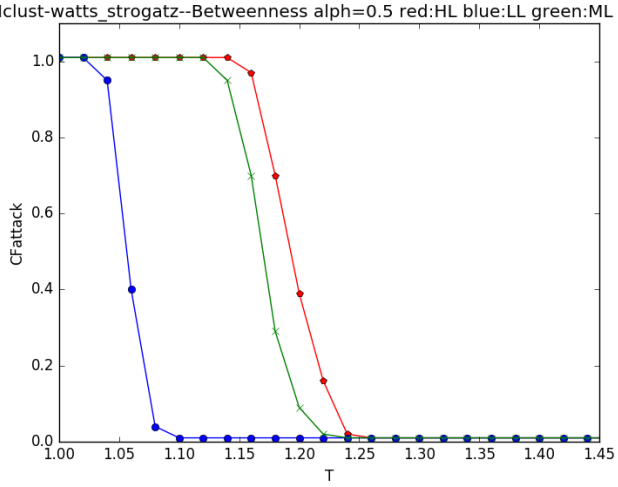
barabasi_albert--Betweenness alph=0.5 red:HL blue:LL green:ML rep20



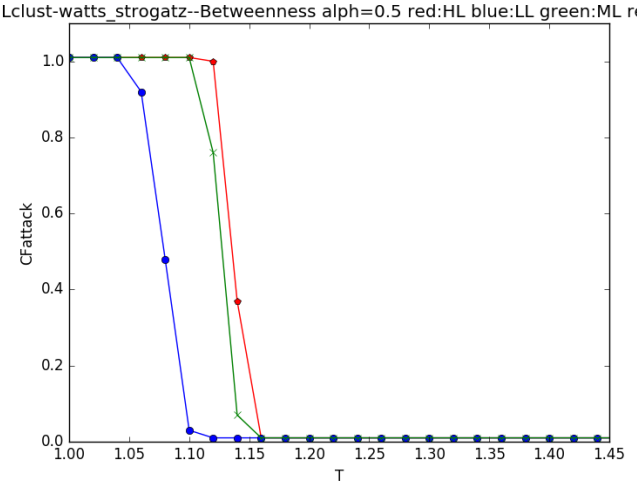
erdos_renyi--Betweenness alph=0.5 red:HL blue:LL green:ML rep20



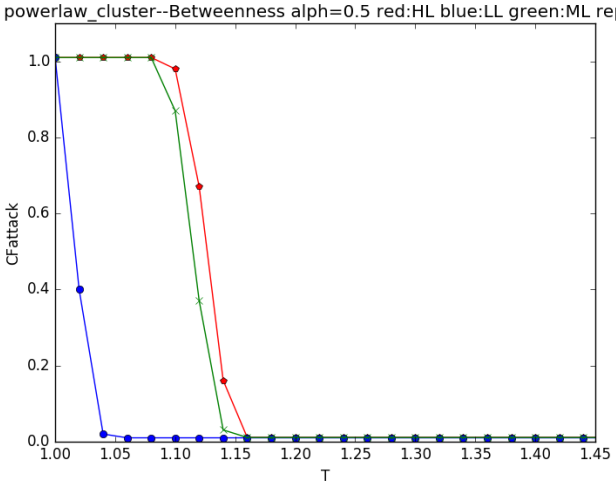
Hclust-watts_strogatz--Betweenness alph=0.5 red:HL blue:LL green:ML rep20



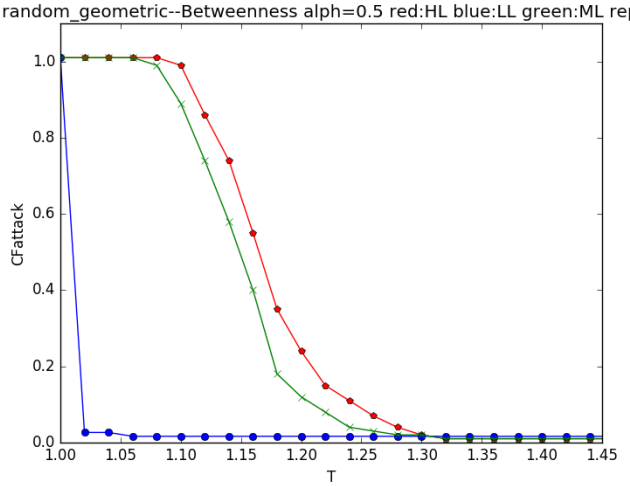
Lclust-watts_strogatz--Betweenness alph=0.5 red:HL blue:LL green:ML rep20



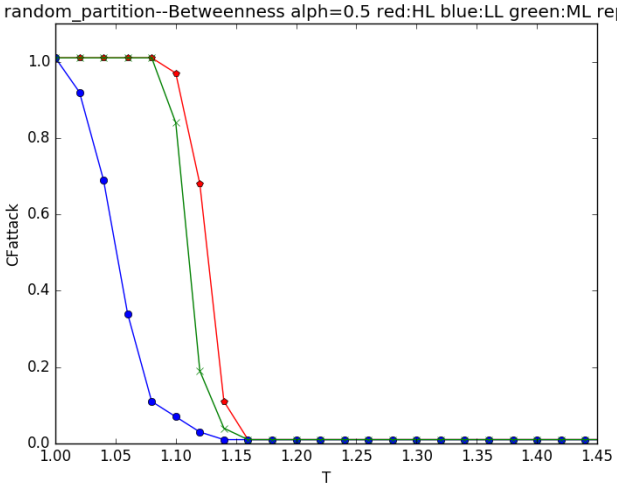
powerlaw_cluster--Betweenness alph=0.5 red:HL blue:LL green:ML rep20



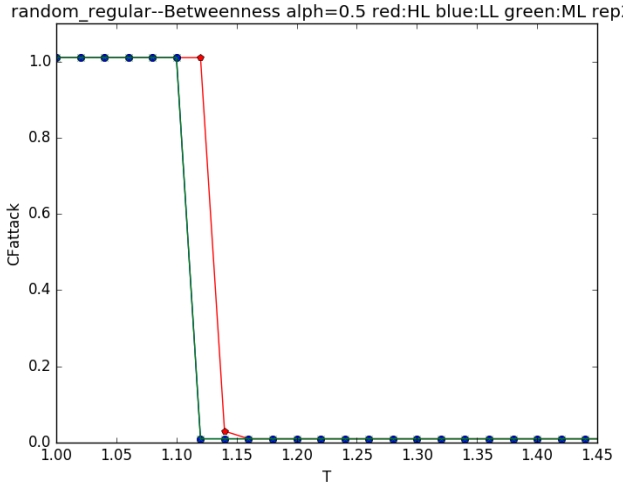
random_geometric--Betweenness alph=0.5 red:HL blue:LL green:ML rep20



random_partition--Betweenness alph=0.5 red:HL blue:LL green:ML rep20

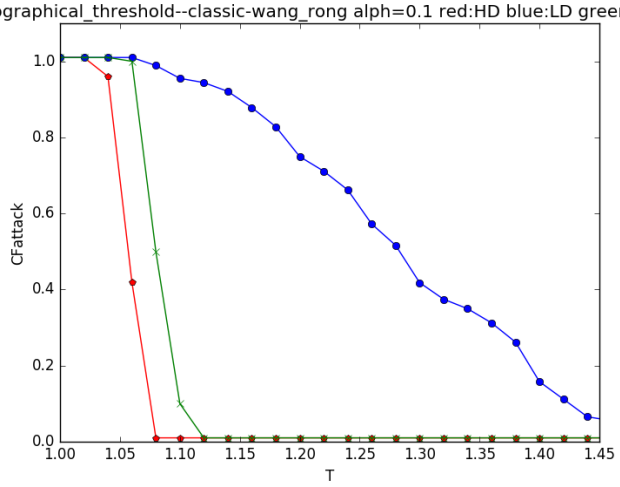


random_regular--Betweenness alph=0.5 red:HL blue:LL green:ML rep20

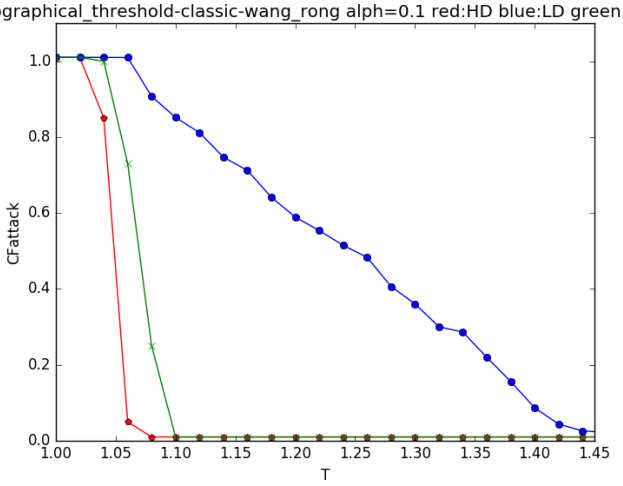


Classic 0.1

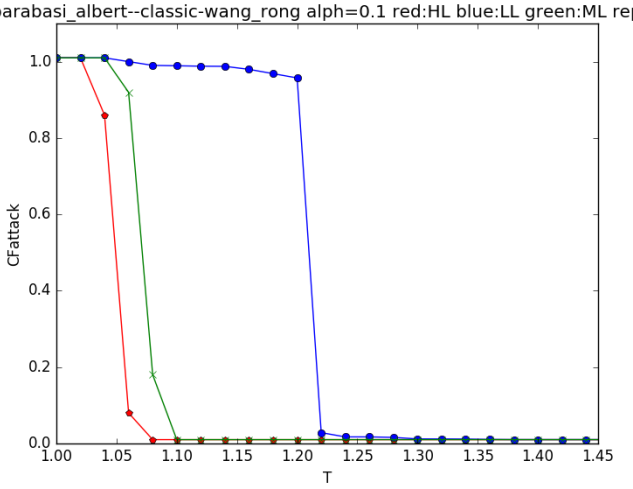
geographical_threshold--classic-wang_rong alph=0.1 red:HD blue:LD green:MD



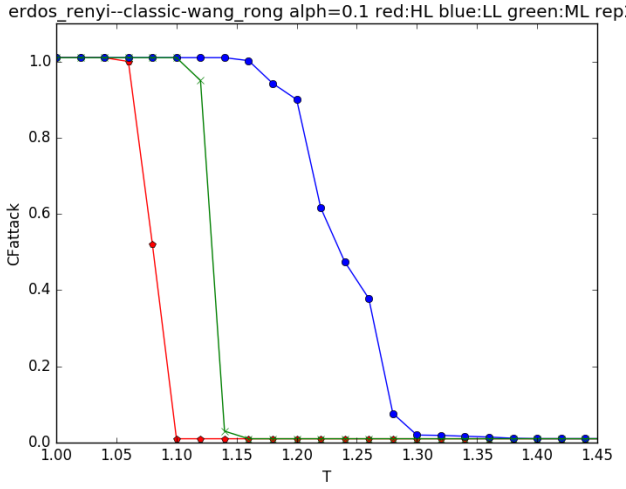
geographical_threshold-classic-wang_rong alph=0.1 red:HD blue:LD green:MD



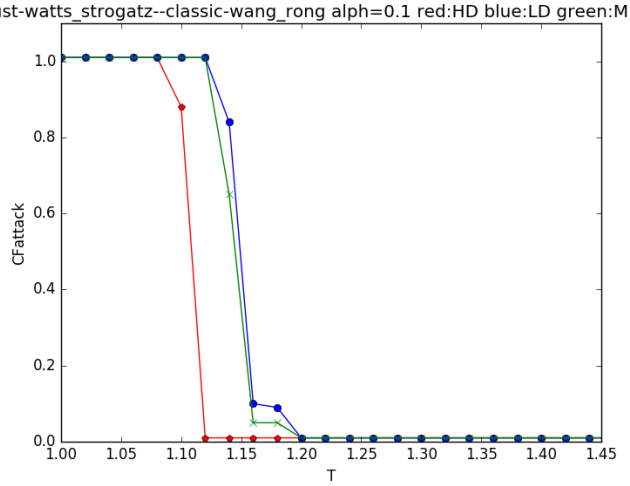
barabasi_albert--classic-wang_rong alph=0.1 red:HL blue:LL green:ML rep20



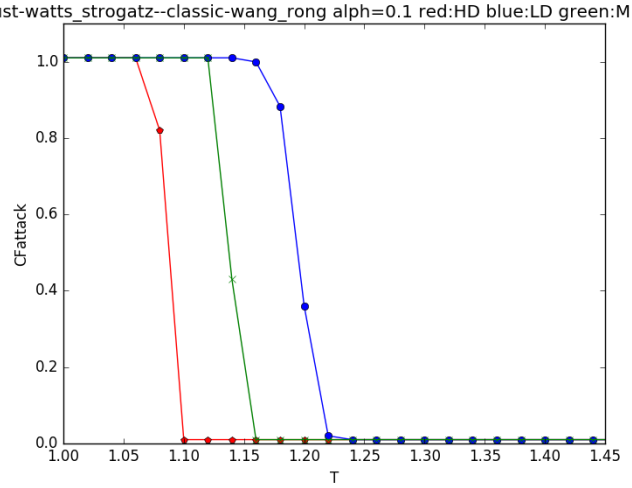
erdos_renyi--classic-wang_rong alph=0.1 red:HL blue:LL green:ML rep20



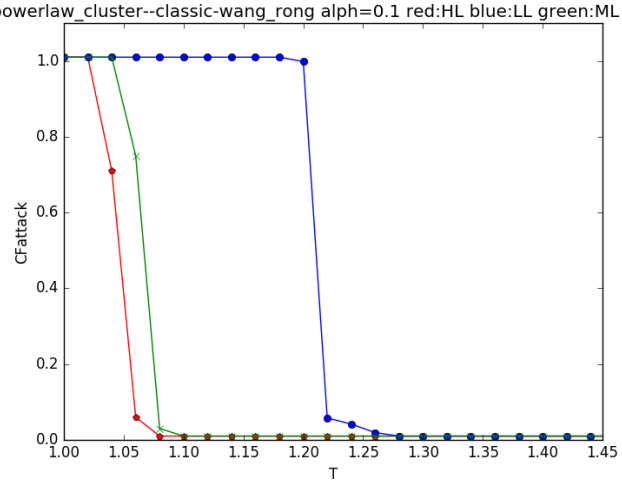
clust-watts_strogatz--classic-wang_rong alph=0.1 red:HD blue:LD green:MD



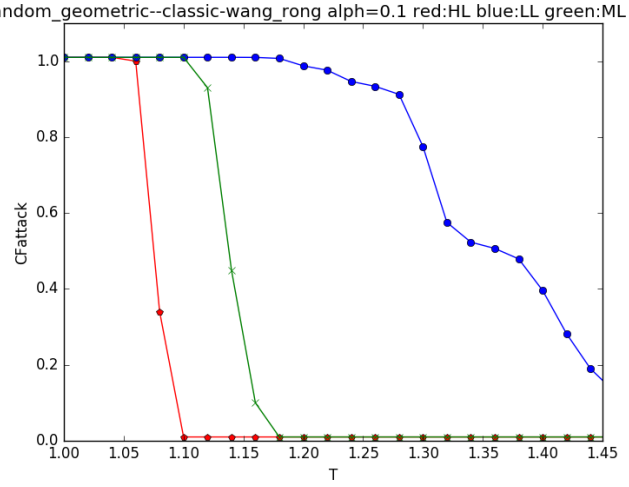
clust-watts_strogatz--classic-wang_rong alph=0.1 red:HD blue:LD green:MD



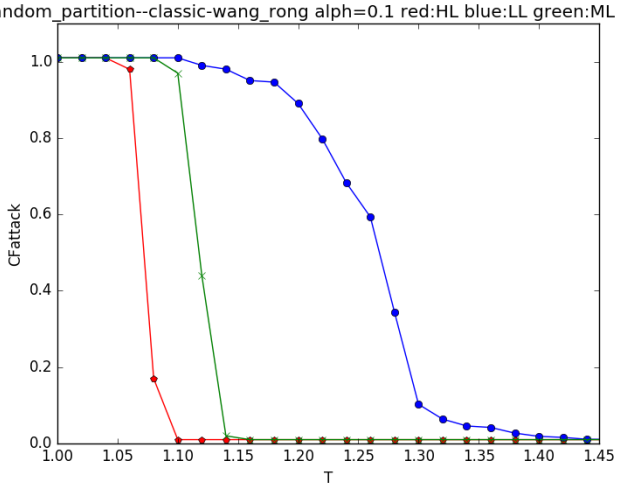
powerlaw_cluster--classic-wang_rong alph=0.1 red:HL blue:LL green:ML rep2(



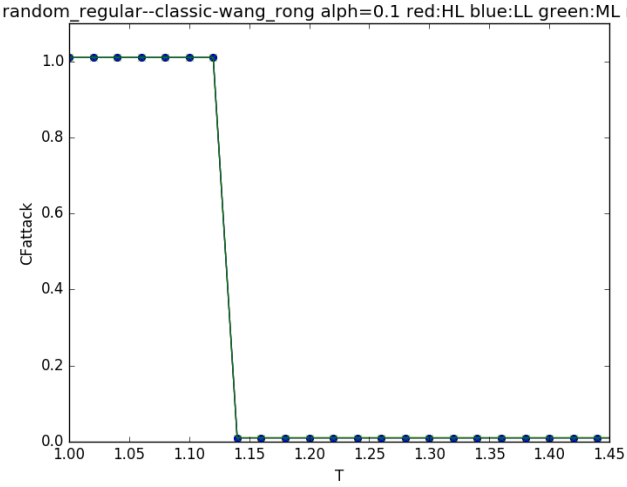
random_geometric--classic-wang_rong alph=0.1 red:HL blue:LL green:ML rep2



random_partition--classic-wang_rong alph=0.1 red:HL blue:LL green:ML rep20

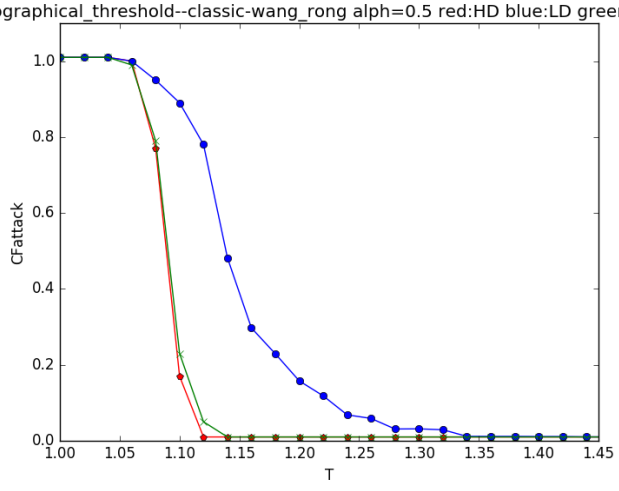


random_regular--classic-wang_rong alph=0.1 red:HL blue:LL green:ML rep20

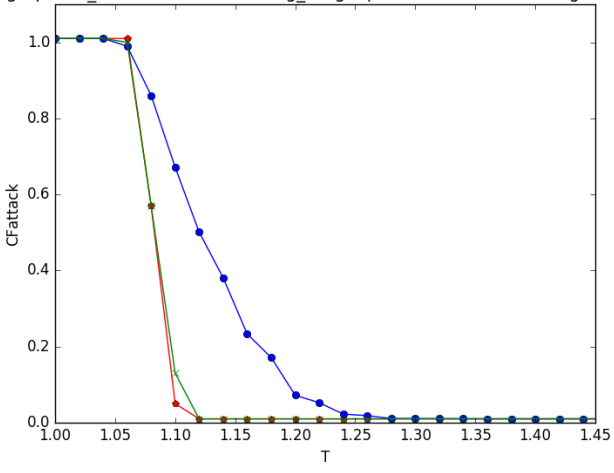


Classic 0.5

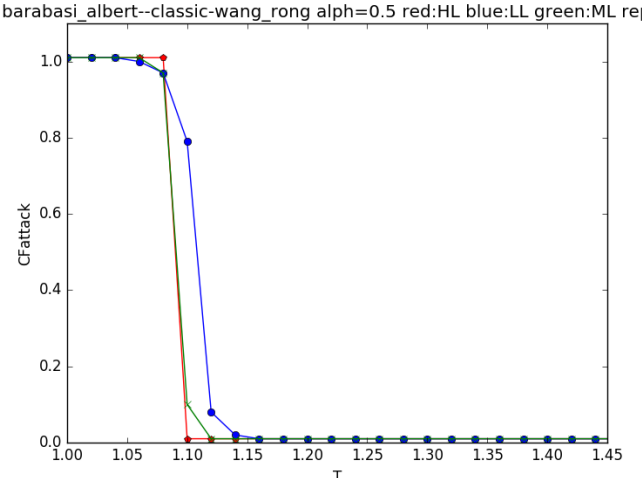
geographical_threshold--classic-wang_rong alph=0.5 red:HD blue:LD green:MD -



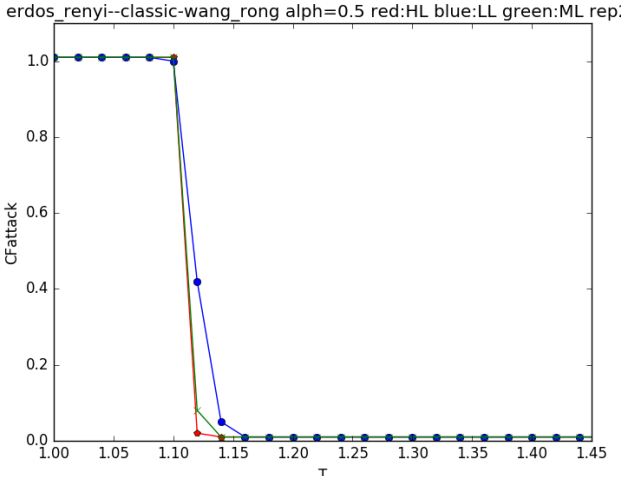
geographical_threshold-classic-wang_rong alph=0.5 red:HL blue:LL green:ML r



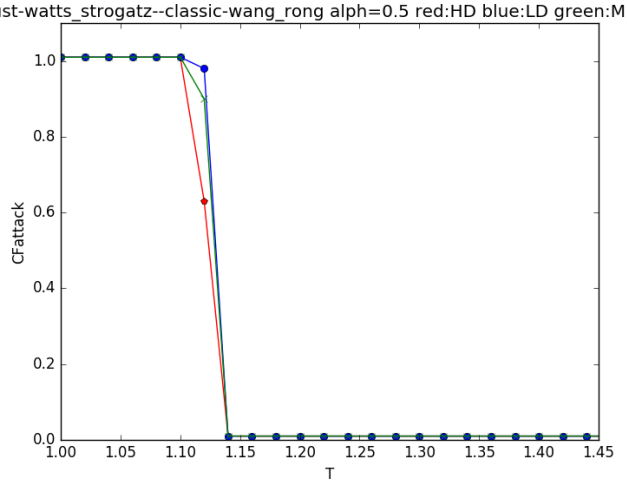
barabasi_albert--classic-wang_rong alph=0.5 red:HL blue:LL green:ML rep20



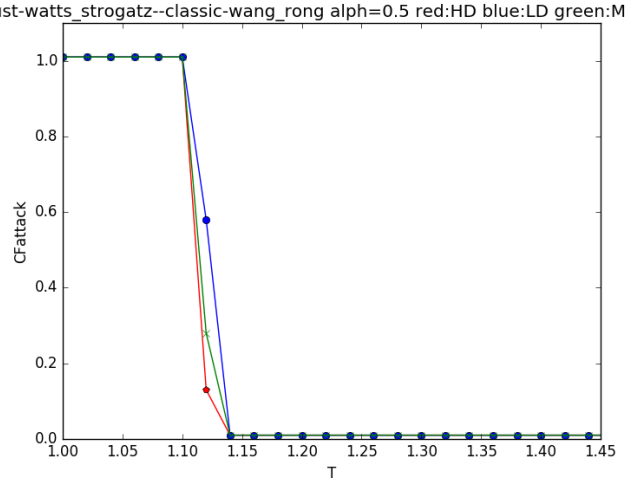
erdos_renyi--classic-wang_rong alph=0.5 red:HL blue:LL green:ML rep20

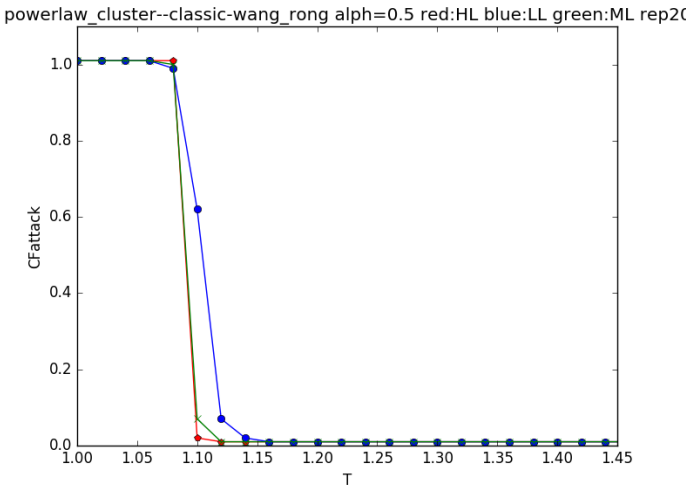


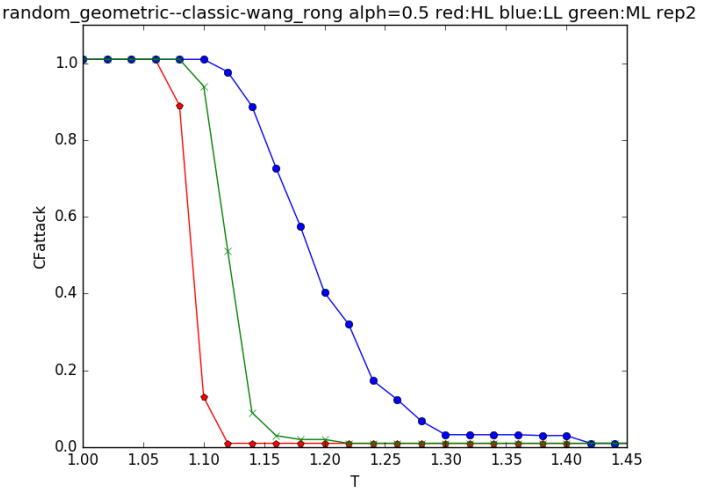
clust-watts_strogatz--classic-wang_rong alph=0.5 red:HD blue:LD green:MD rep

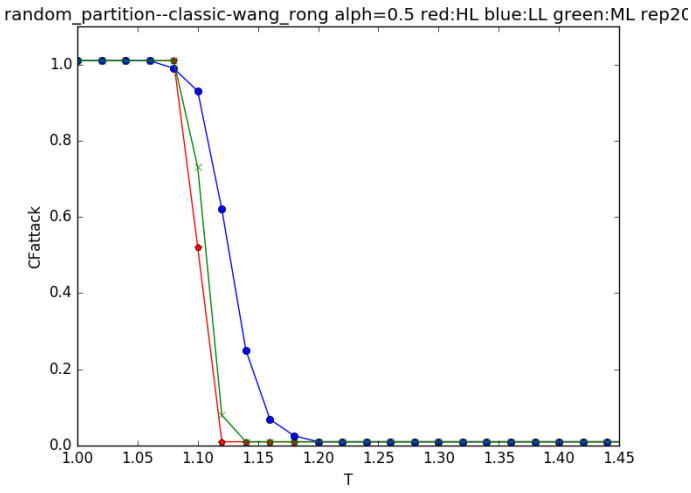


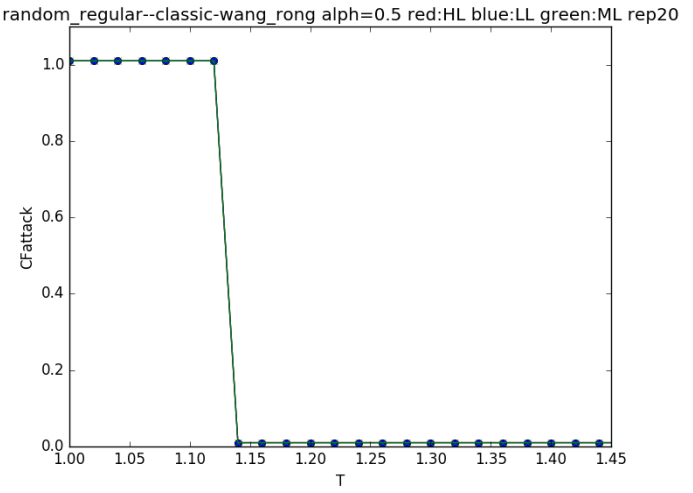
clust-watts_strogatz--classic-wang_rong alph=0.5 red:HD blue:LD green:MD rep



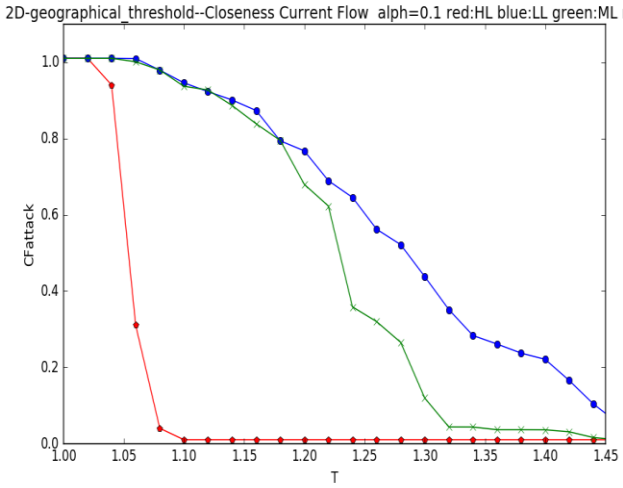


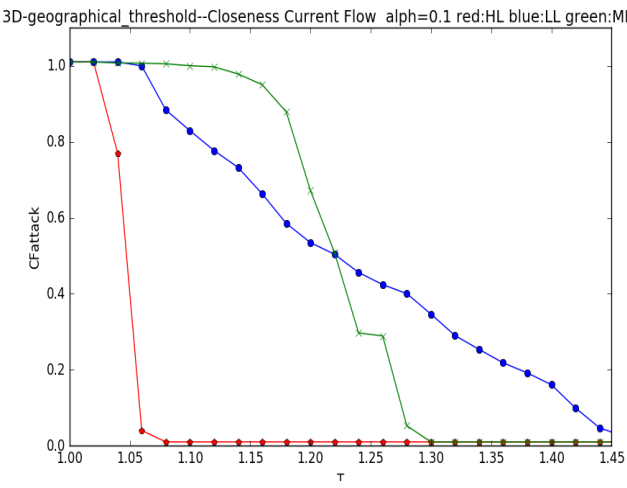


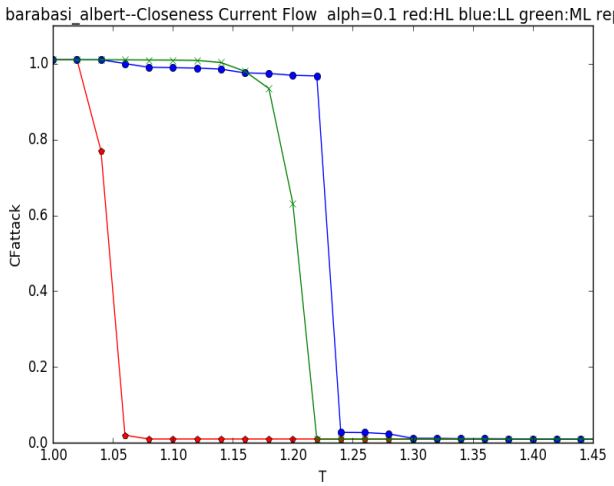


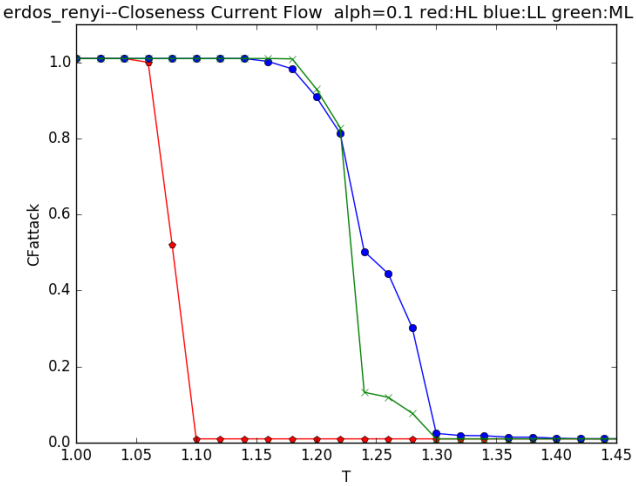


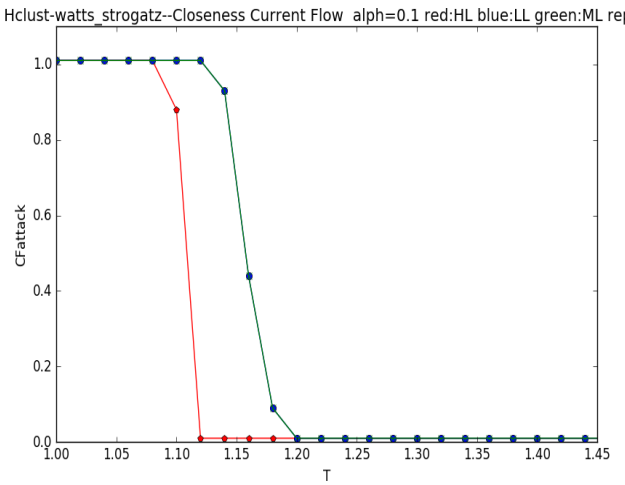
Closseness 0.1

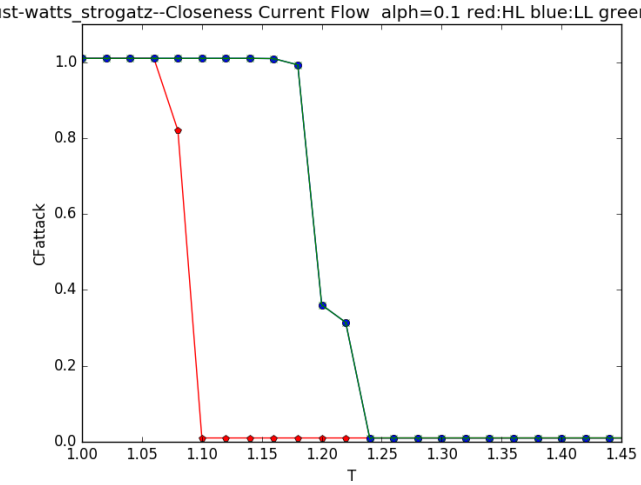

2D-geographical_threshold--Closeness Current Flow alph=0.1 red:HL blue:LL green:ML rep20
CFattack
T


3D-geographical_threshold--Closeness Current Flow alph=0.1 red:HL blue:LL green:ML rep20
CFattack
T

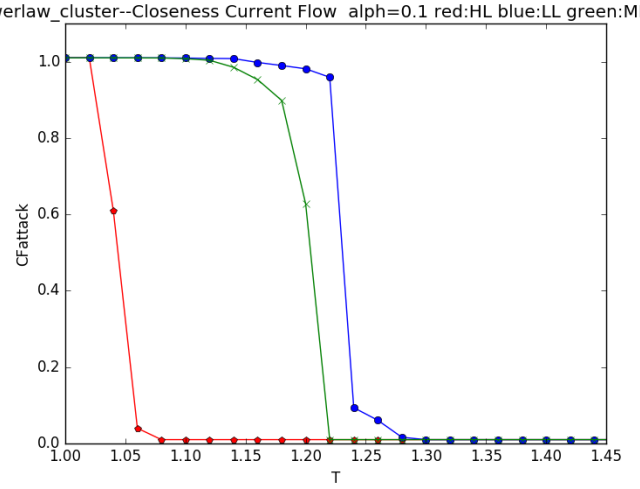

barabasi_albert--Closeness Current Flow alph=0.1 red:HL blue:LL green:ML rep20
CFattack
T


erdos_renyi--Closeness Current Flow alph=0.1 red:HL blue:LL green:ML rep20
CFattack
T

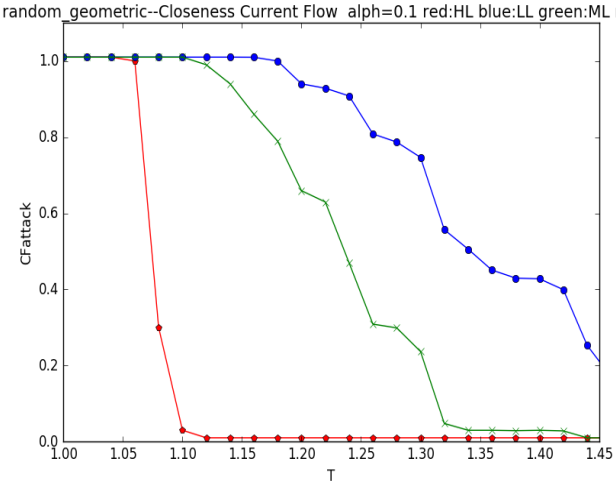

Hclust-watts_strogatz--Closeness Current Flow alph=0.1 red:HL blue:LL green:ML rep20
CFattack
T


ust-watts_strogatz--Closeness Current Flow alph=0.1 red:HL blue:LL green:ML r
CFattack
T

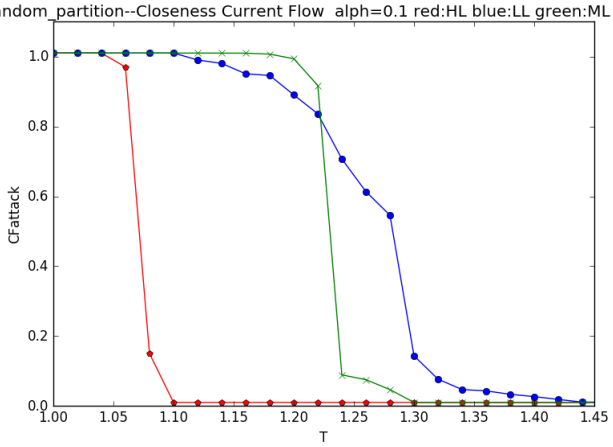
owerlaw_cluster--Closeness Current Flow alph=0.1 red:HL blue:LL green:ML re



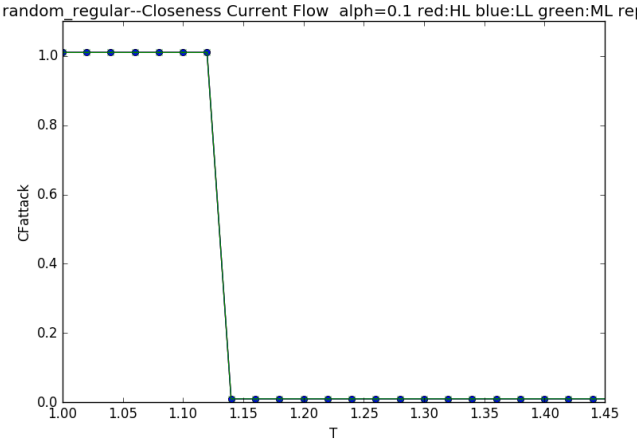
random_geometric--Closeness Current Flow alph=0.1 red:HL blue:LL green:ML rep20



random_partition--Closeness Current Flow alph=0.1 red:HL blue:LL green:ML rep20

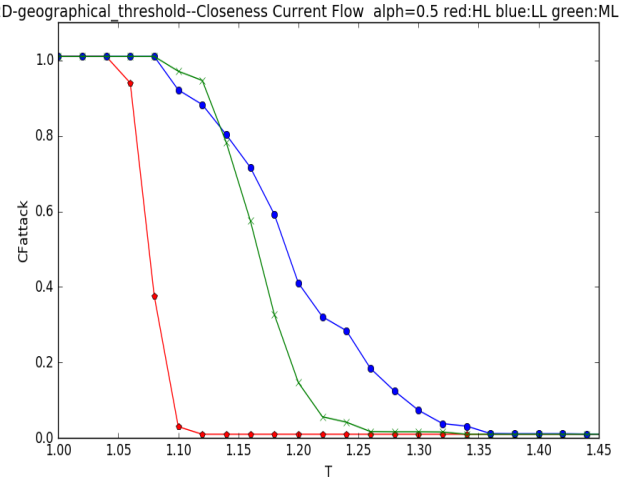


random_regular--Closeness Current Flow alph=0.1 red:HL blue:LL green:ML rep20

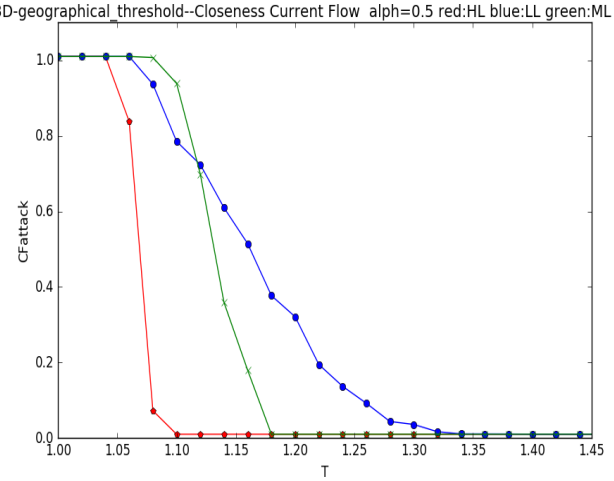


Closseness 0.5

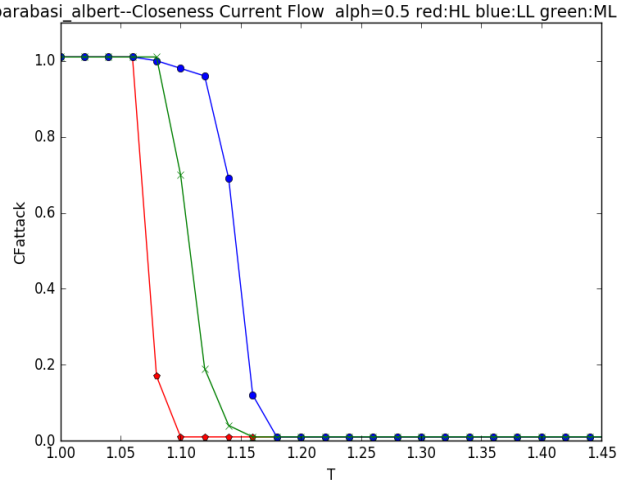
2D-geographical_threshold--Closeness Current Flow alph=0.5 red:HL blue:LL green:ML rep20



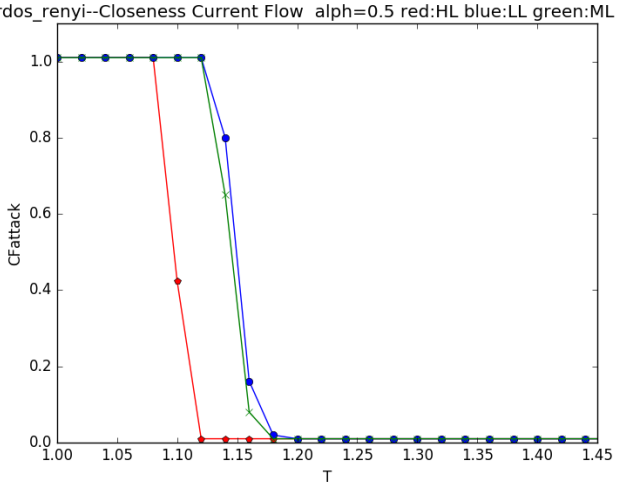
3D-geographical_threshold--Closeness Current Flow alph=0.5 red:HL blue:LL green:ML rep20



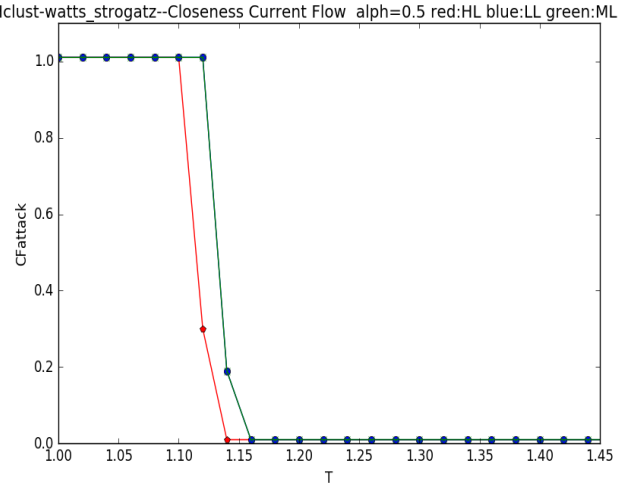
barabasi_albert--Closeness Current Flow alph=0.5 red:HL blue:LL green:ML rep20



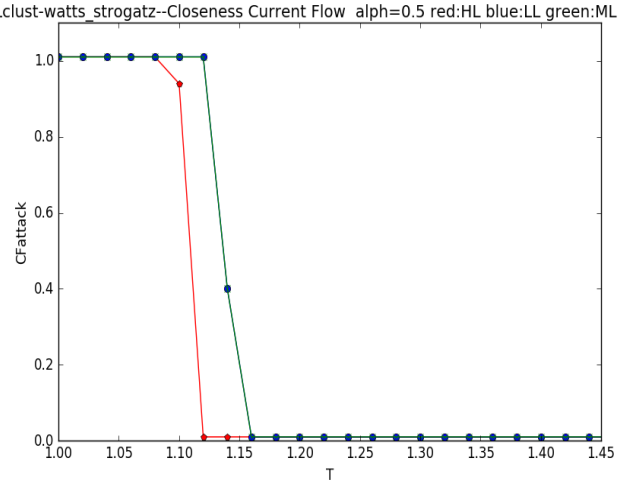
erdos_renyi--Closeness Current Flow alph=0.5 red:HL blue:LL green:ML rep20

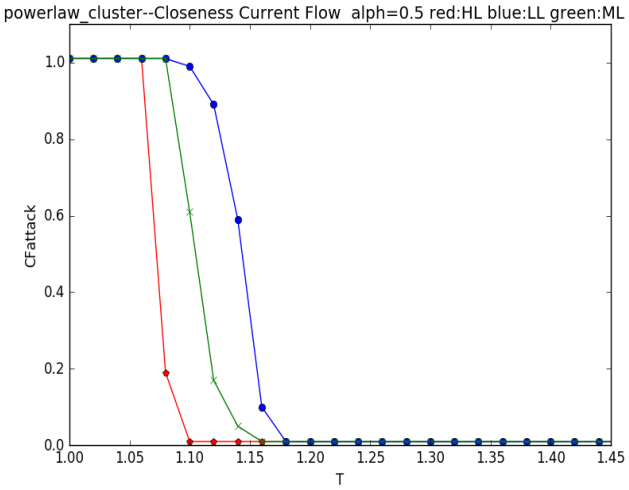


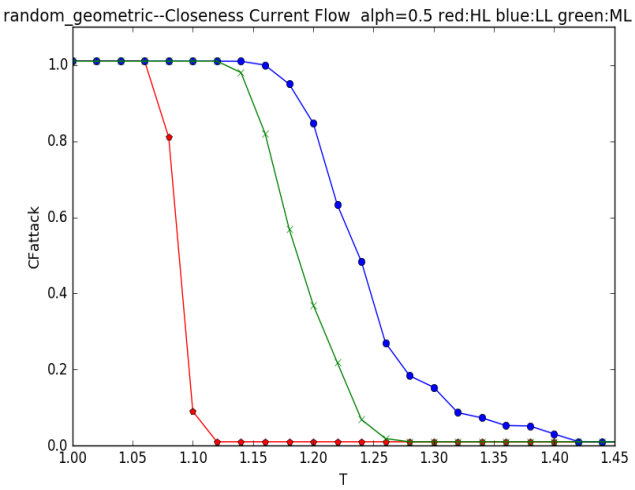
Hclust-watts_strogatz--Closeness Current Flow alph=0.5 red:HL blue:LL green:ML rep20

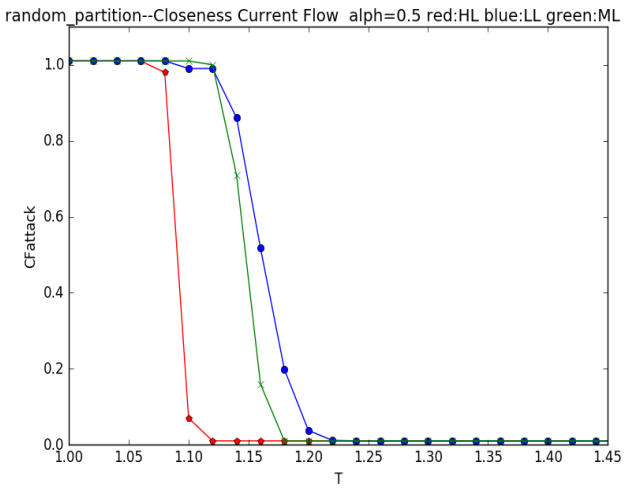


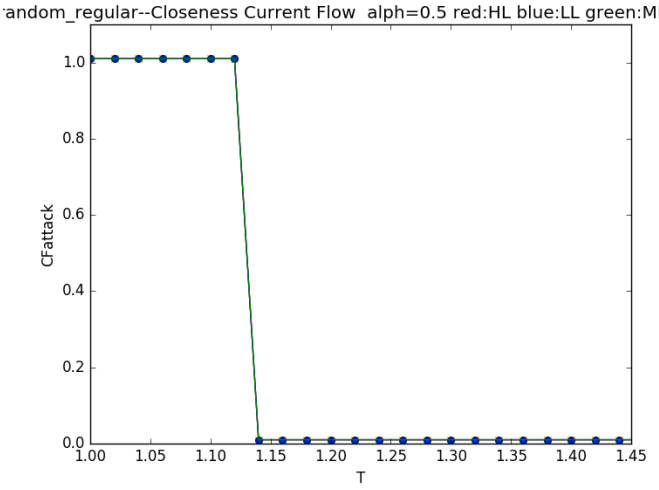
Lclust-watts_strogatz--Closeness Current Flow alph=0.5 red:HL blue:LL green:ML rep20





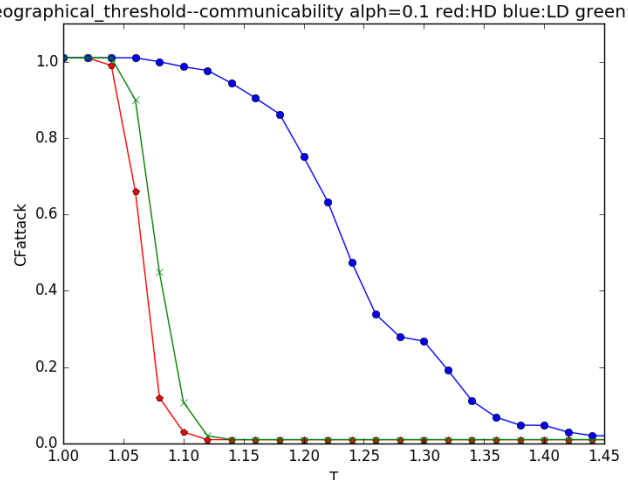




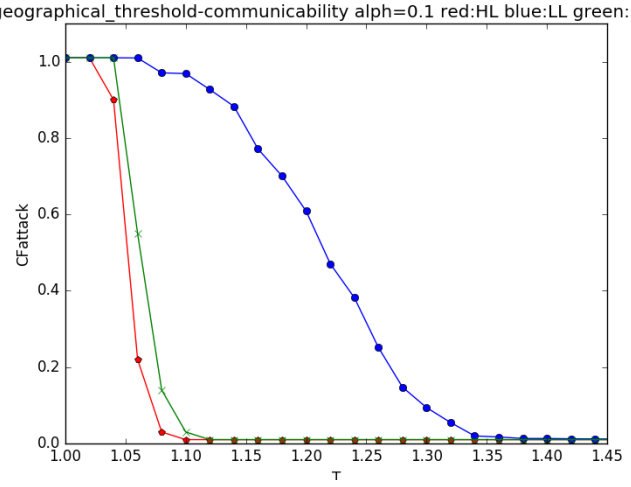


Communicability 0.1

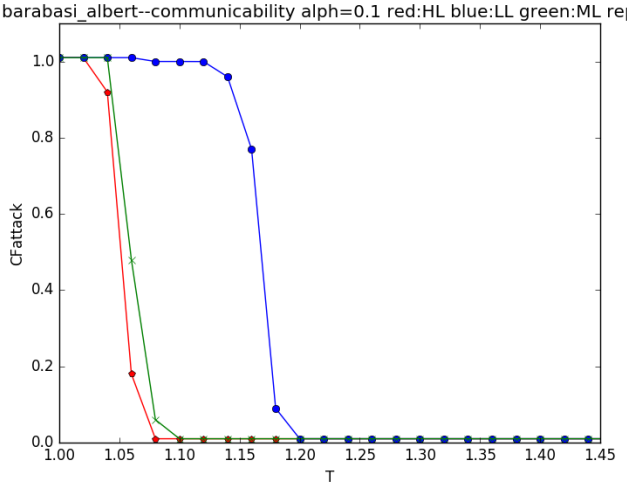
-geographical_threshold--communicability alph=0.1 red:HD blue:LD green:MD r



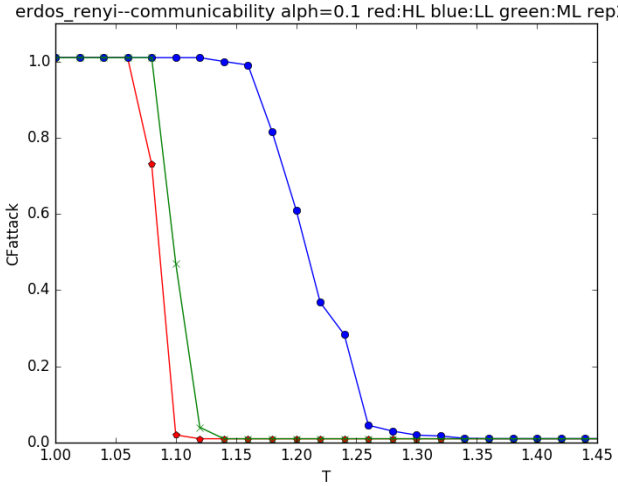
)-geographical_threshold-communicability alph=0.1 red:HL blue:LL green:ML re



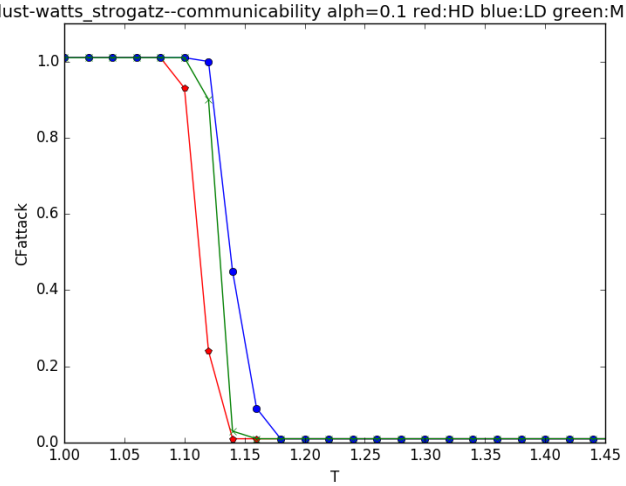
barabasi_albert--communicability alph=0.1 red:HL blue:LL green:ML rep20



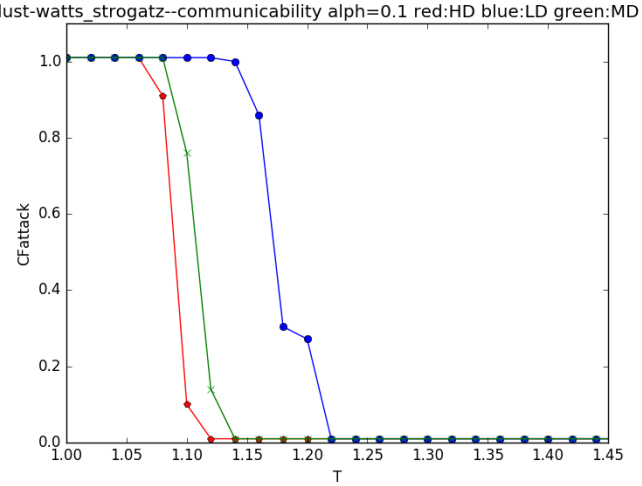
erdos_renyi--communicability alph=0.1 red:HL blue:LL green:ML rep20

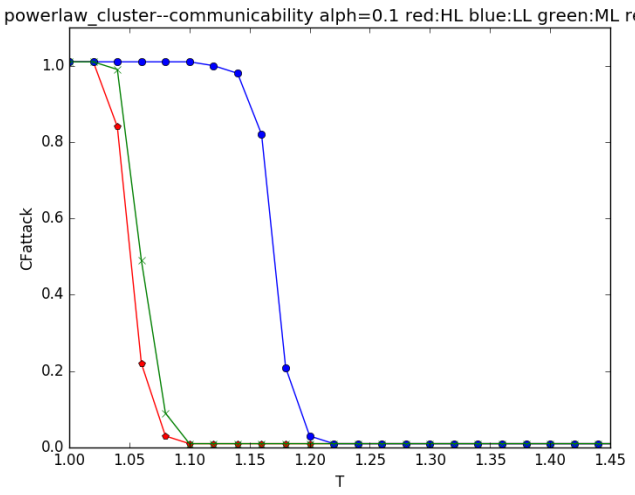


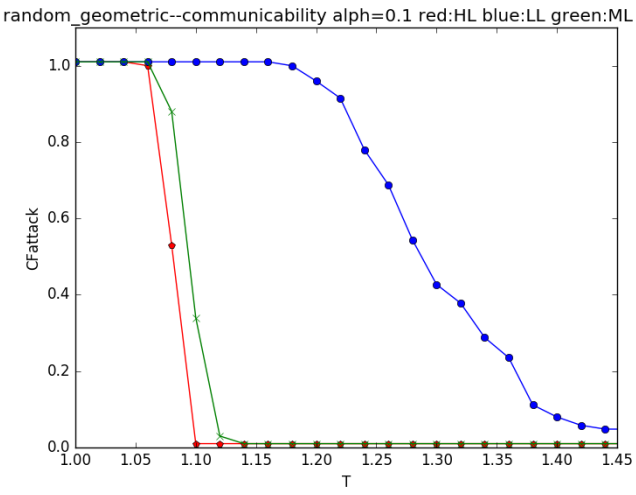
Hclust-watts_strogatz--communicability alph=0.1 red:HD blue:LD green:MD rep

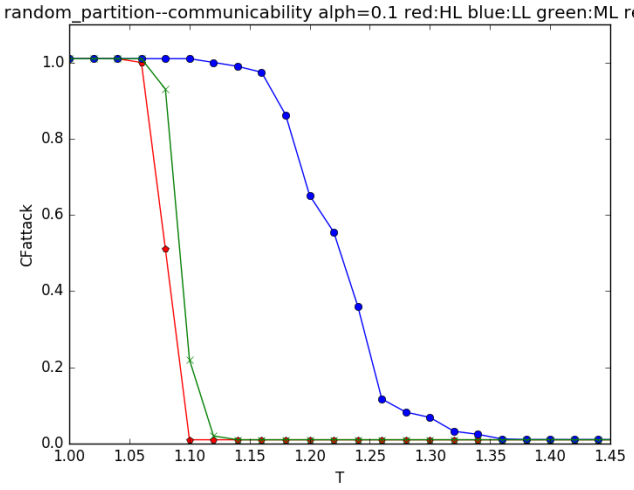


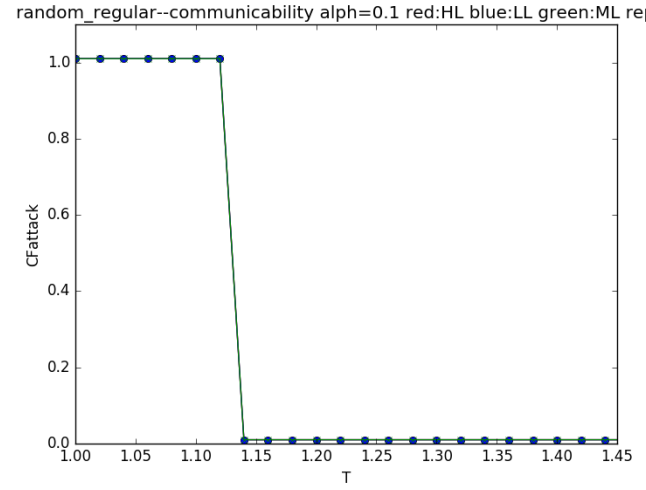
Lclust-watts_strogatz--communicability alph=0.1 red:HD blue:LD green:MD rep



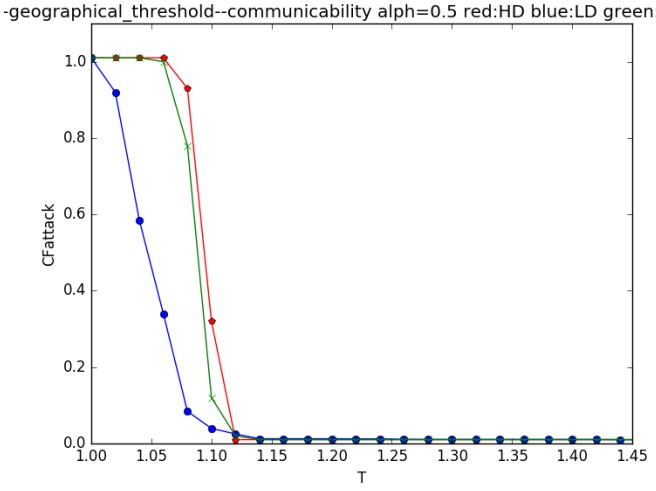


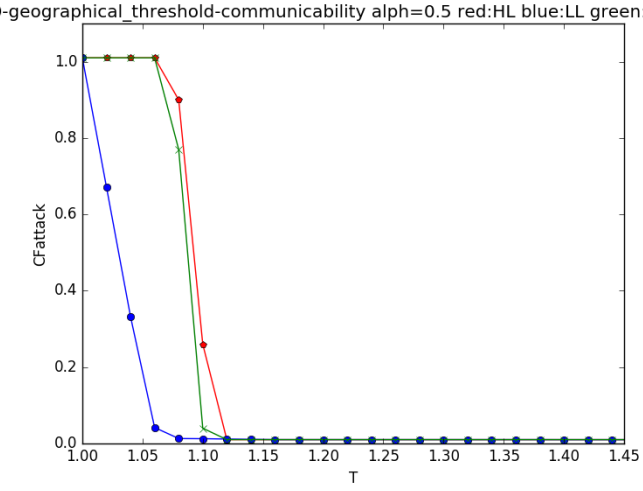




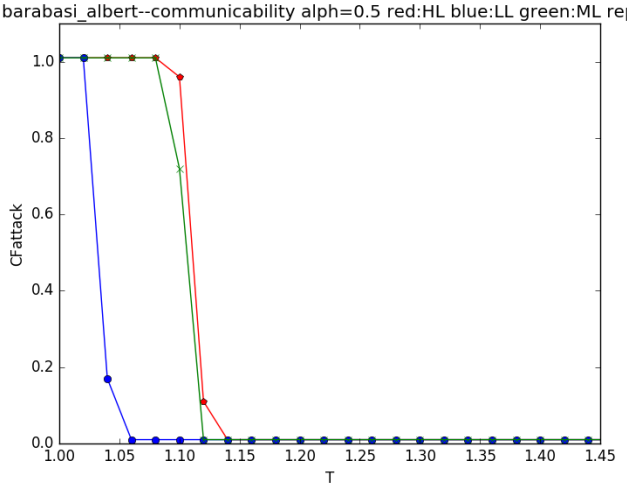


Communicability 0.5

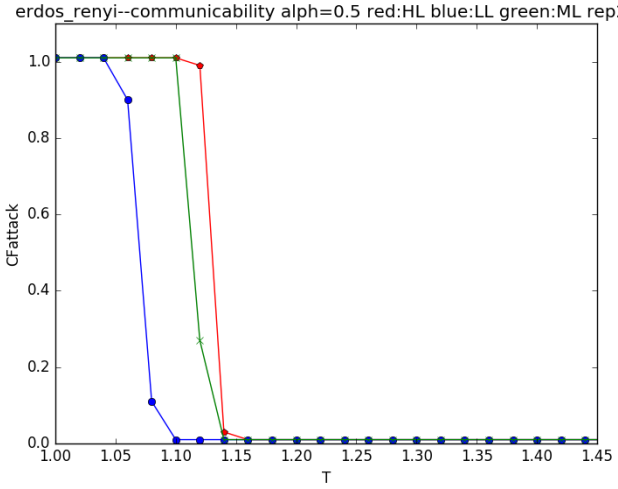




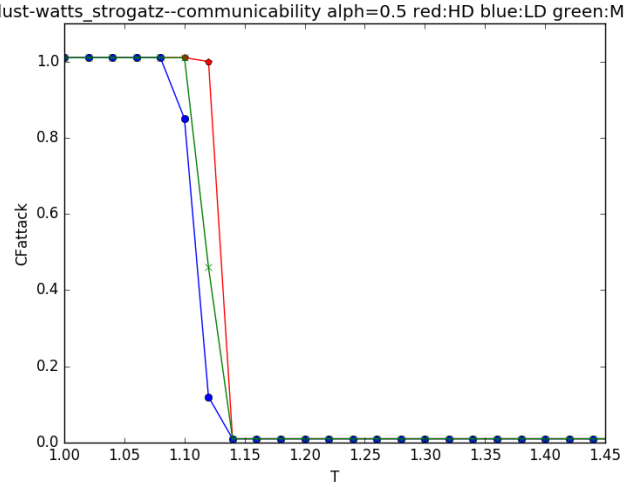
barabasi_albert--communicability alph=0.5 red:HL blue:LL green:ML rep20

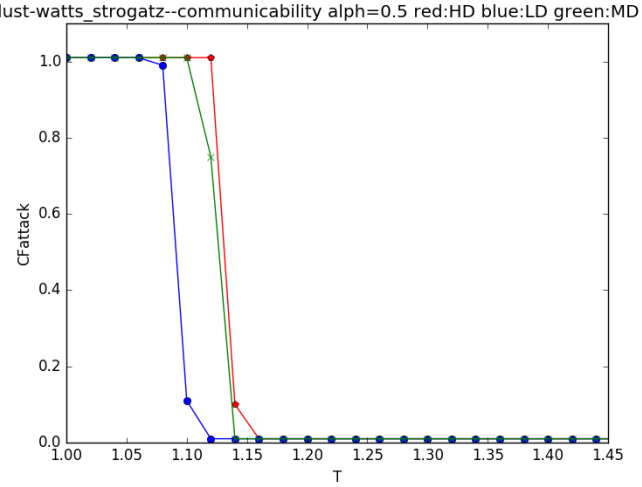


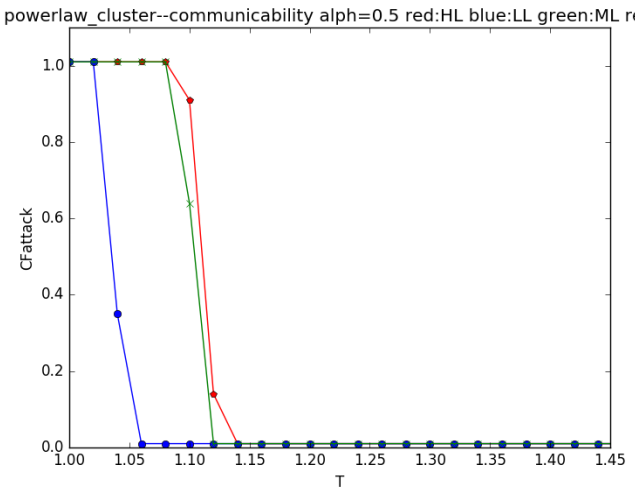
erdos_renyi--communicability alph=0.5 red:HL blue:LL green:ML rep20

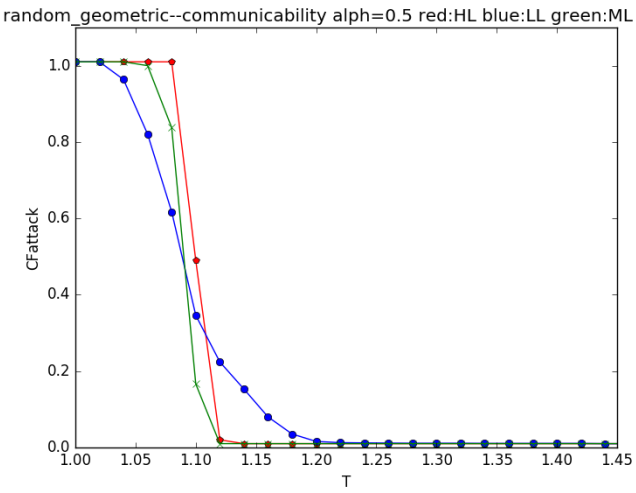


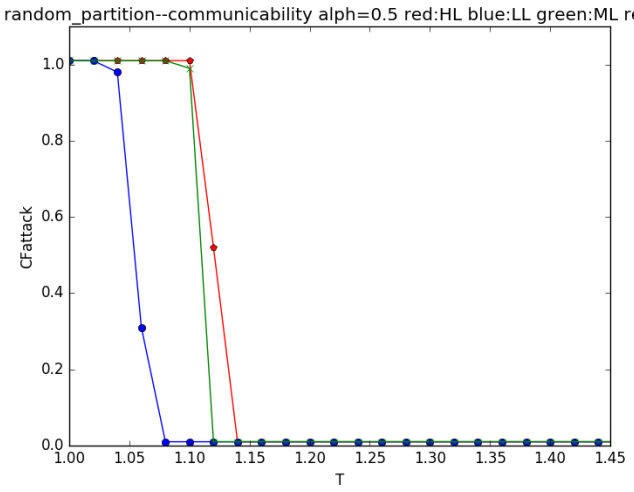
Hclust-watts_strogatz--communicability alph=0.5 red:HD blue:LD green:MD repLclust-watts_strogatz--communicability alph=0.5 red:HD blue:LD green:MD rep2

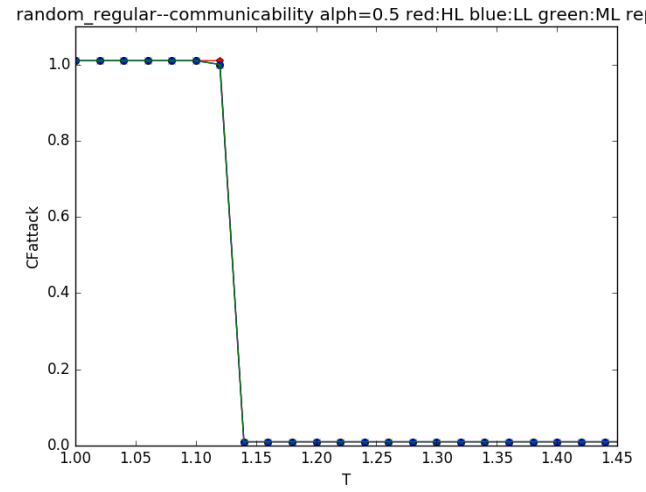






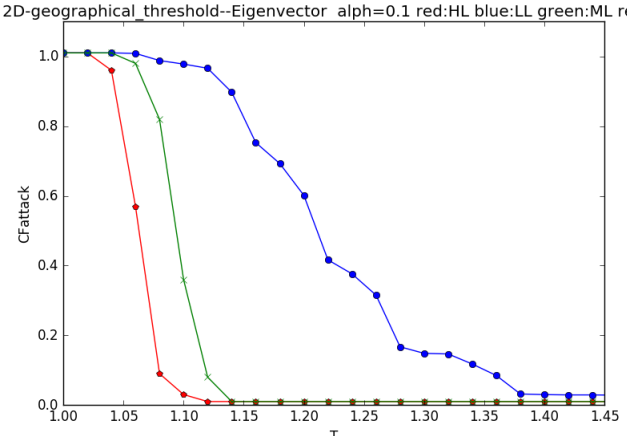




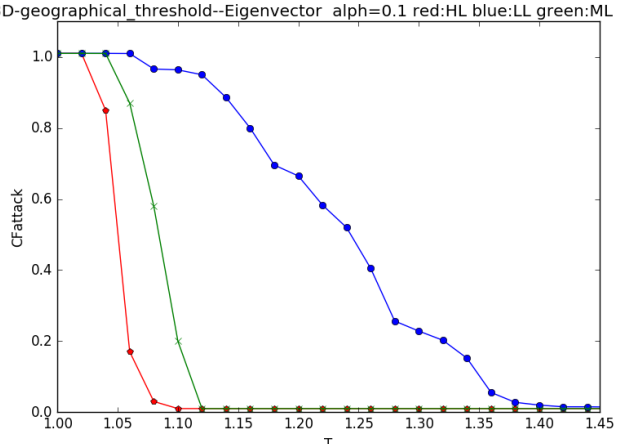


Eigenvector 0.1

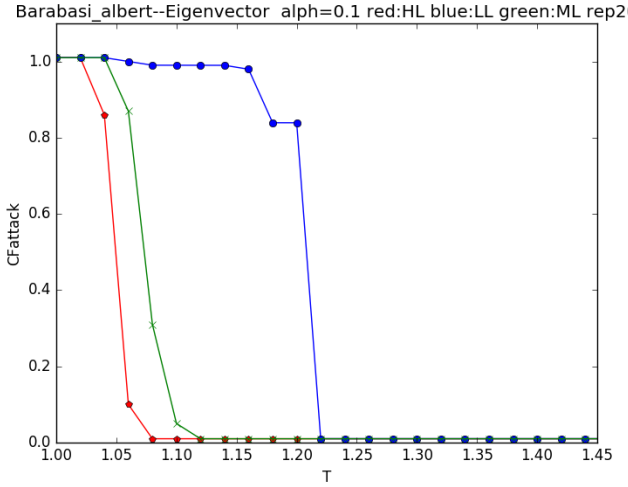
2D-geographical_threshold--Eigenvector alph=0.1 red:HL blue:LL green:ML rep20



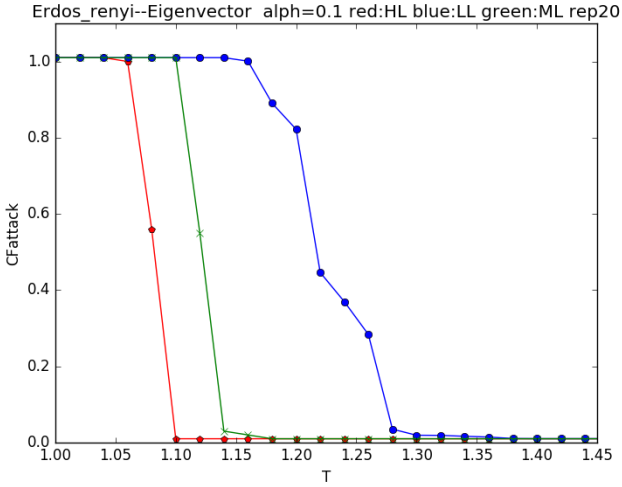
3D-geographical_threshold--Eigenvector alph=0.1 red:HL blue:LL green:ML rep20



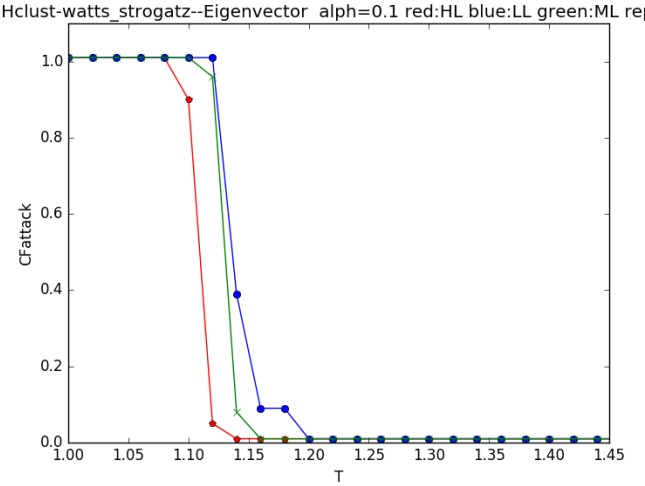
Barabasi_albert--Eigenvector alph=0.1 red:HL blue:LL green:ML rep20



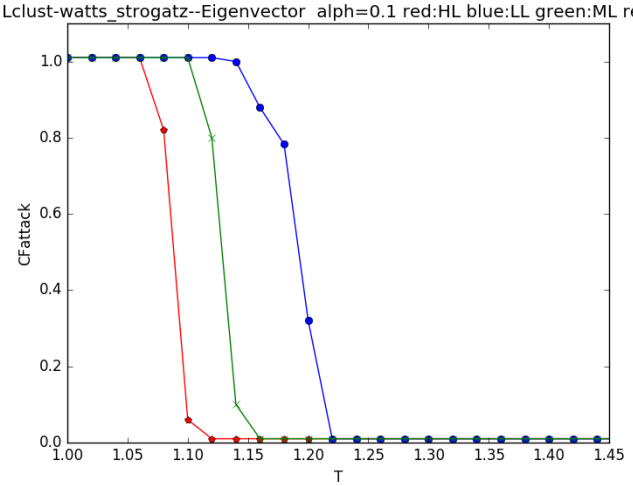
Erdos_renyi--Eigenvector alph=0.1 red:HL blue:LL green:ML rep20

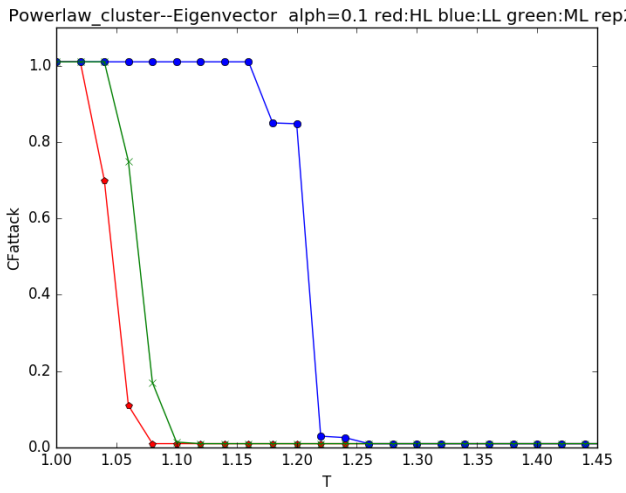


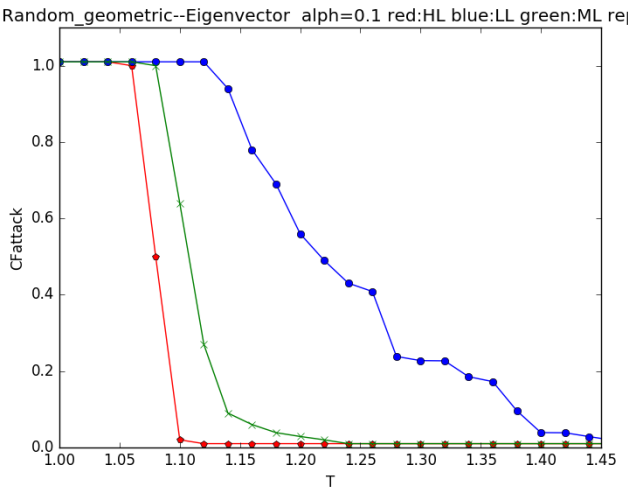
Hclust-watts_strogatz--Eigenvector alph=0.1 red:HL blue:LL green:ML rep20

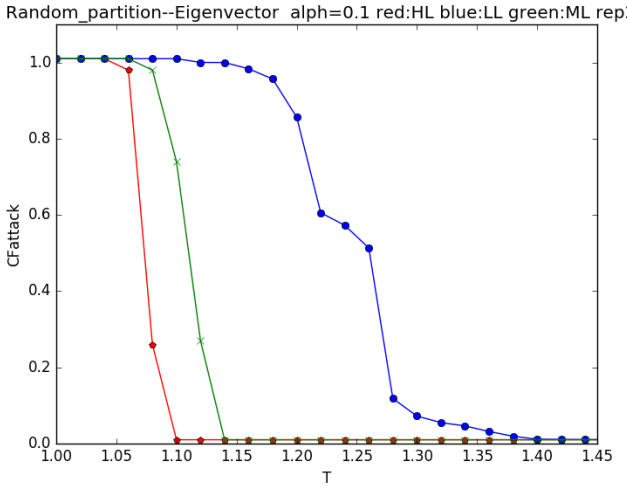


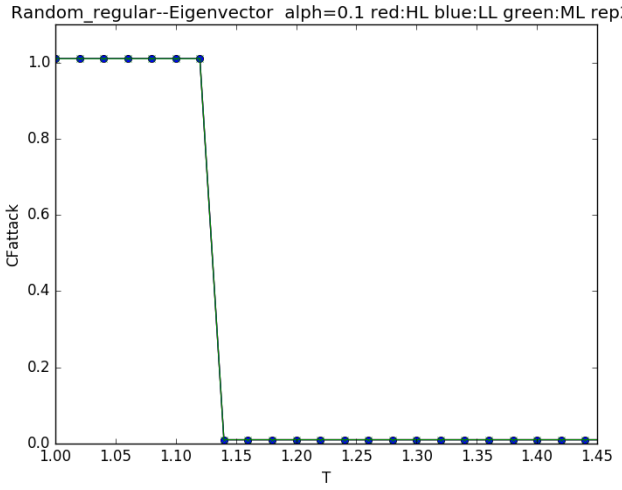
Lclust-watts_strogatz--Eigenvector alph=0.1 red:HL blue:LL green:ML rep20



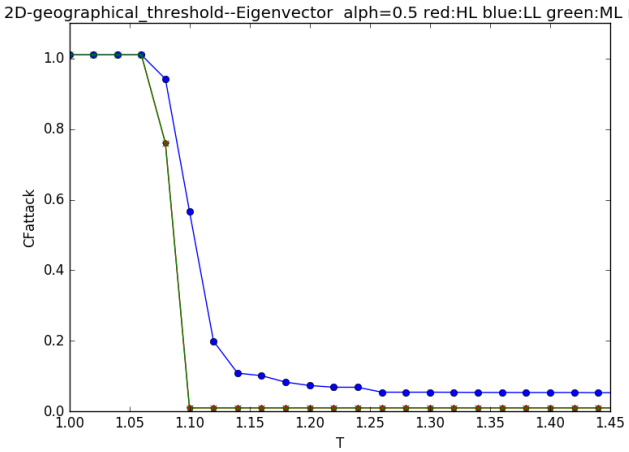


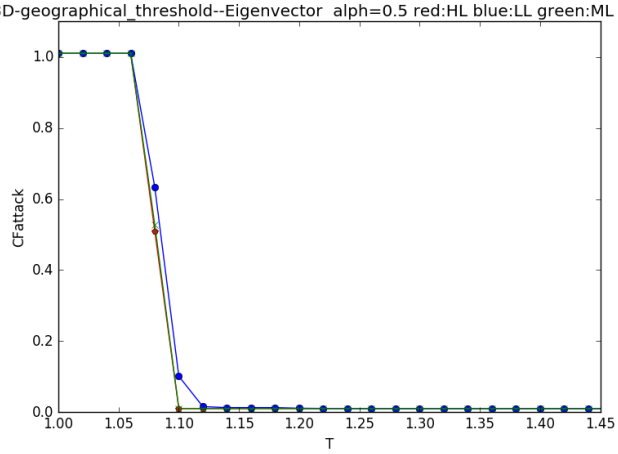


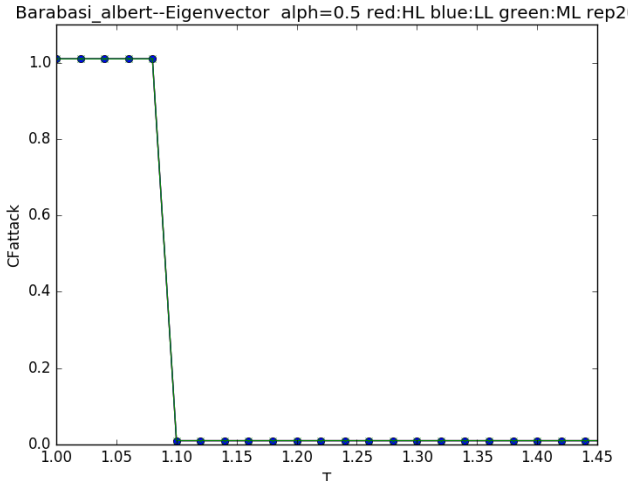


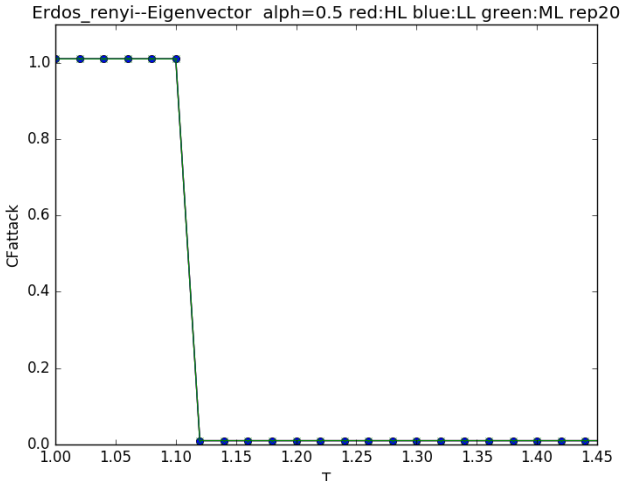


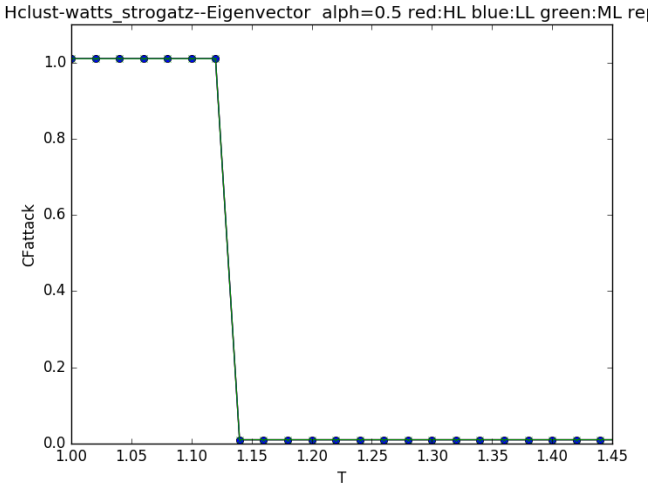
Eigenvector 0.5

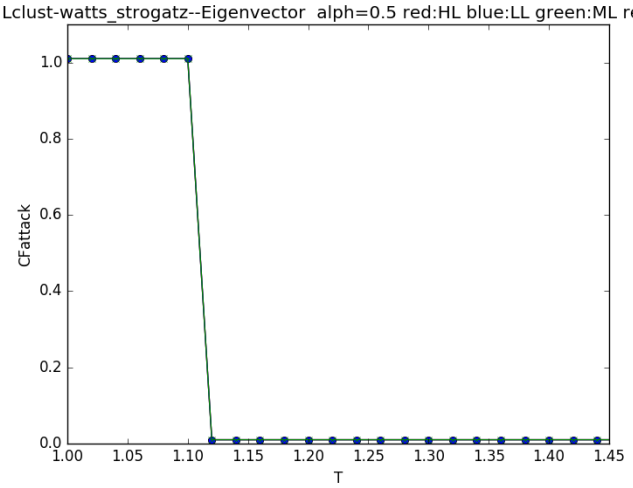


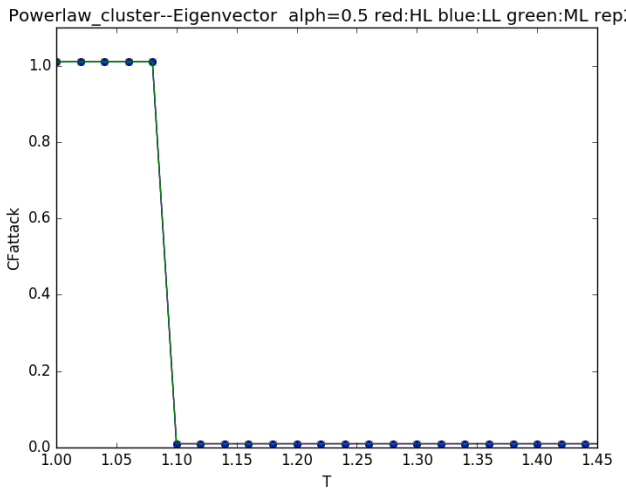


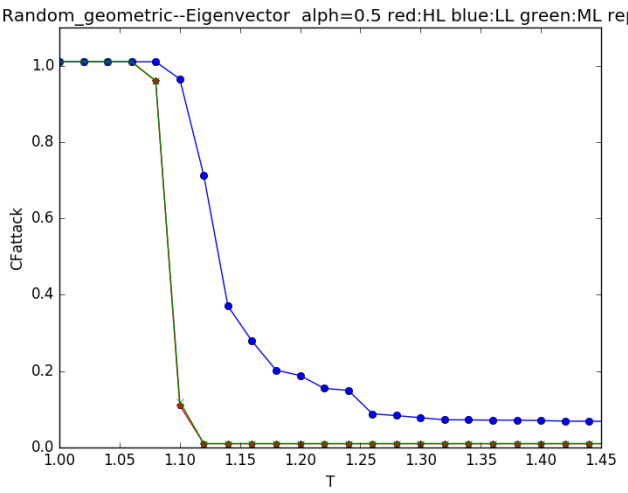

Barabasi_albert--Eigenvector alph=0.5 red:HL blue:LL green:ML rep20
CFattack
T

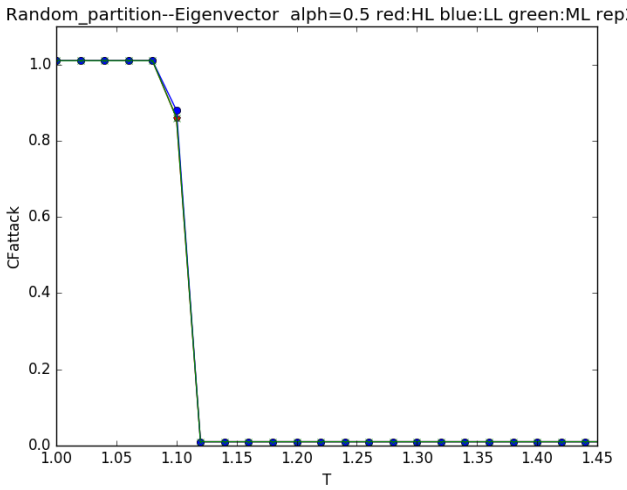

Erdos_renyi--Eigenvector alph=0.5 red:HL blue:LL green:ML rep20
CFattack
T

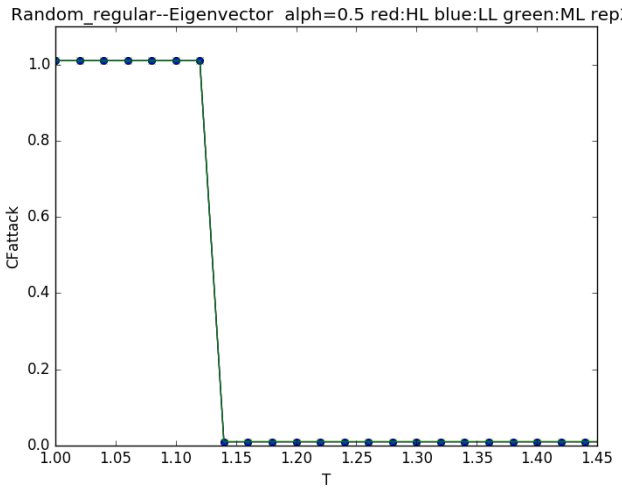

Hclust-watts_strogatz--Eigenvector alph=0.5 red:HL blue:LL green:ML rep20
CFattack
T


Lclust-watts_strogatz--Eigenvector alph=0.5 red:HL blue:LL green:ML rep20
CFattack
T


Powerlaw_cluster--Eigenvector alph=0.5 red:HL blue:LL green:ML rep20
CFattack
T

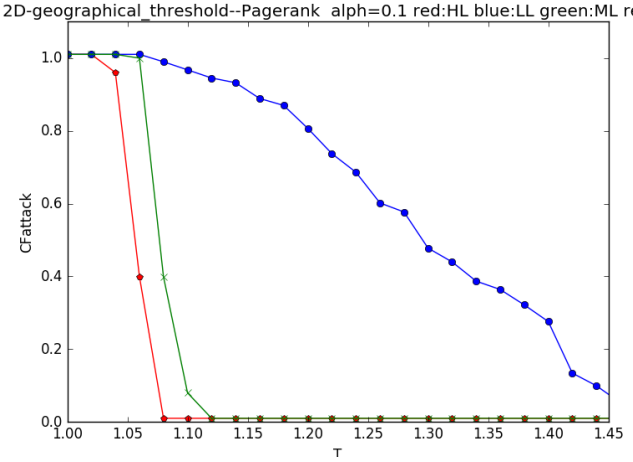

Random_geometric--Eigenvector alph=0.5 red:HL blue:LL green:ML rep20
CFattack
T


Random_partition--Eigenvector alph=0.5 red:HL blue:LL green:ML rep20
CFattack
T

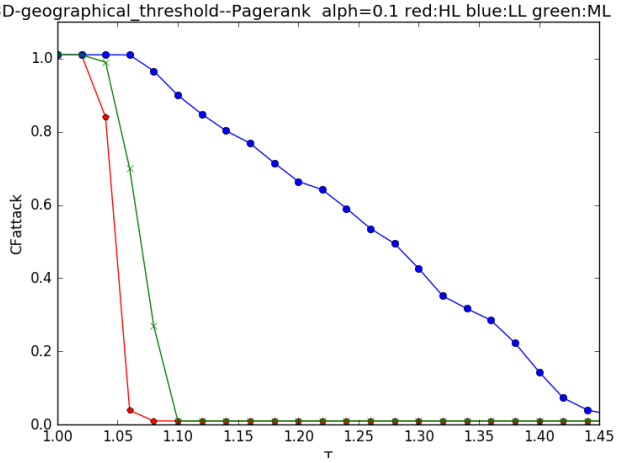

Random_regular--Eigenvector alph=0.5 red:HL blue:LL green:ML rep20
CFattack
T

PageRank 0.1

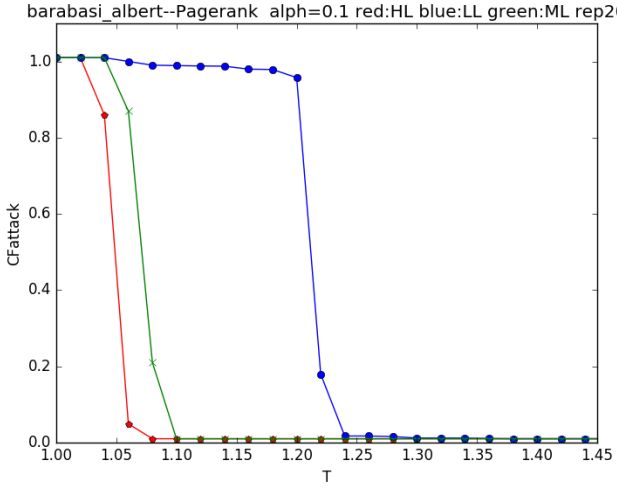
2D-geographical_threshold--Pagerank alph=0.1 red:HL blue:LL green:ML rep20



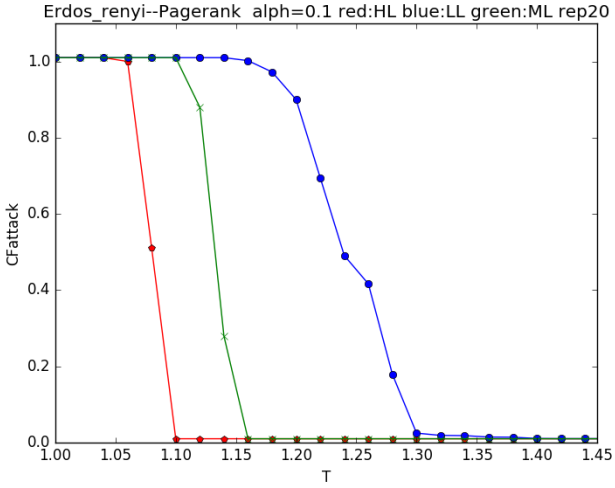
3D-geographical_threshold--Pagerank alph=0.1 red:HL blue:LL green:ML rep20



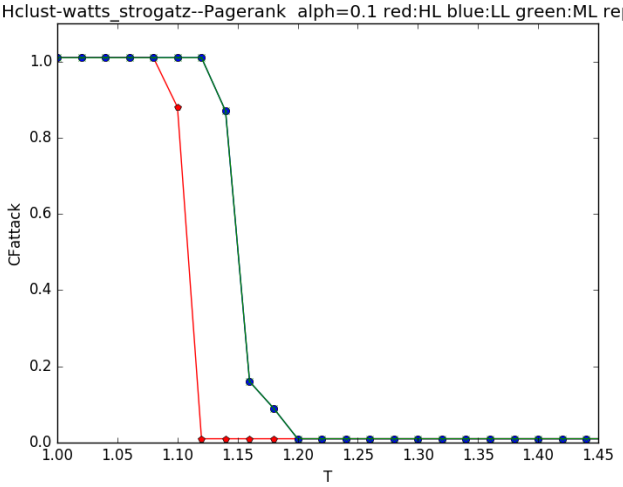
barabasi_albert--Pagerank alph=0.1 red:HL blue:LL green:ML rep20



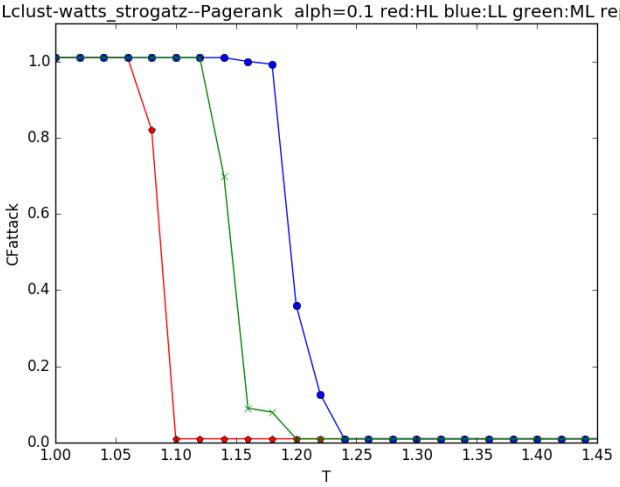
Erdos_renyi--Pagerank alph=0.1 red:HL blue:LL green:ML rep20

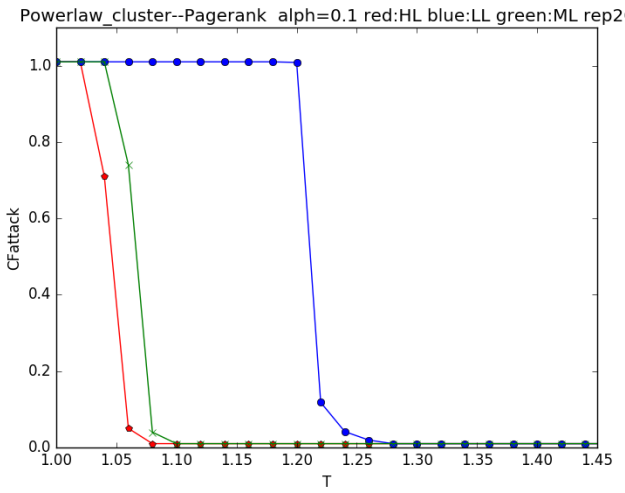


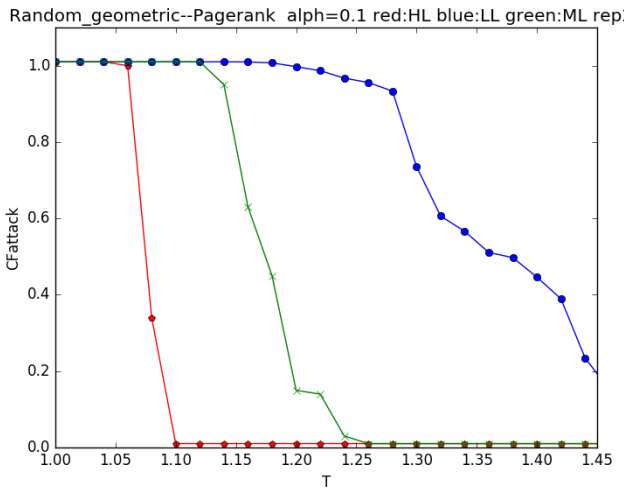
Hclust-watts_strogatz--Pagerank alph=0.1 red:HL blue:LL green:ML rep20

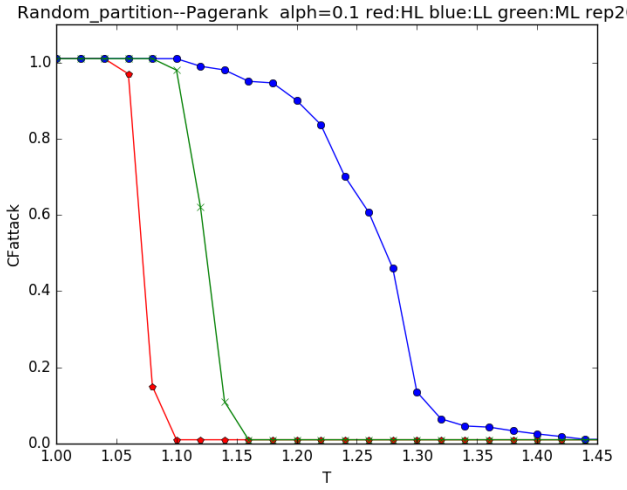


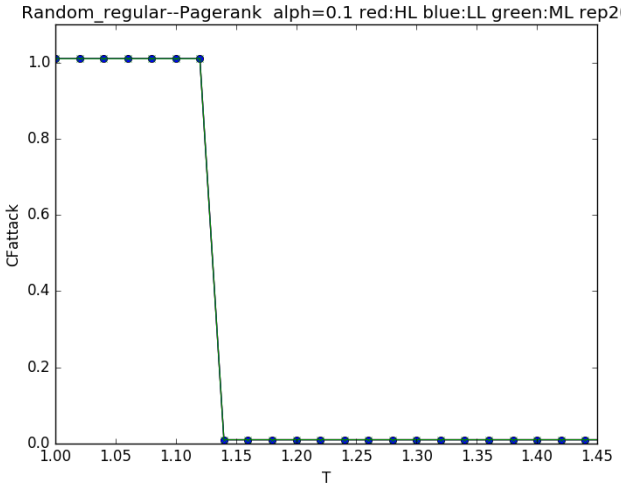
Lclust-watts_strogatz--Pagerank alph=0.1 red:HL blue:LL green:ML rep20



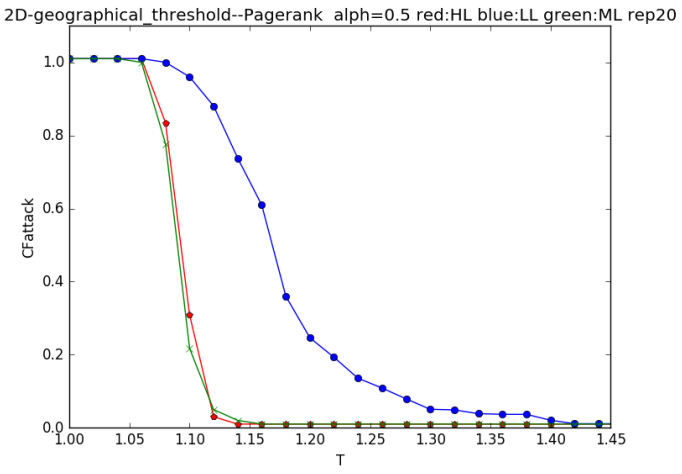

Powerlaw_cluster--Pagerank alph=0.1 red:HL blue:LL green:ML rep20
CFattack
T

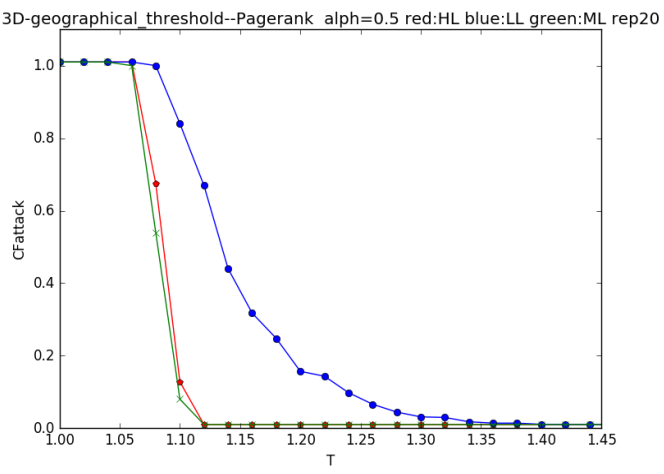

Random_geometric--Pagerank alph=0.1 red:HL blue:LL green:ML rep20
CFattack
T

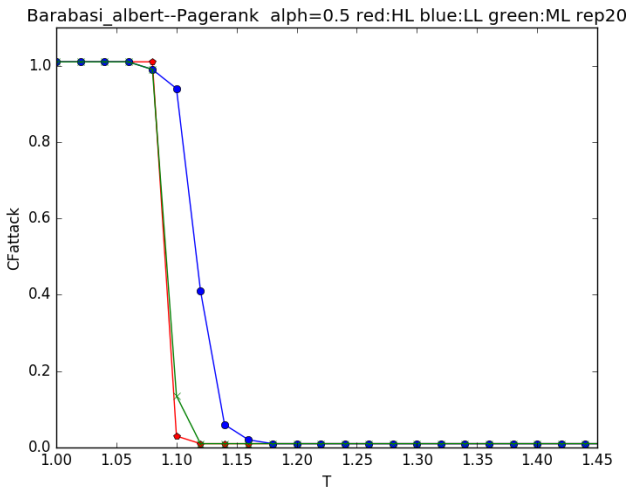

Random_partition--Pagerank alph=0.1 red:HL blue:LL green:ML rep20
CFattack
T

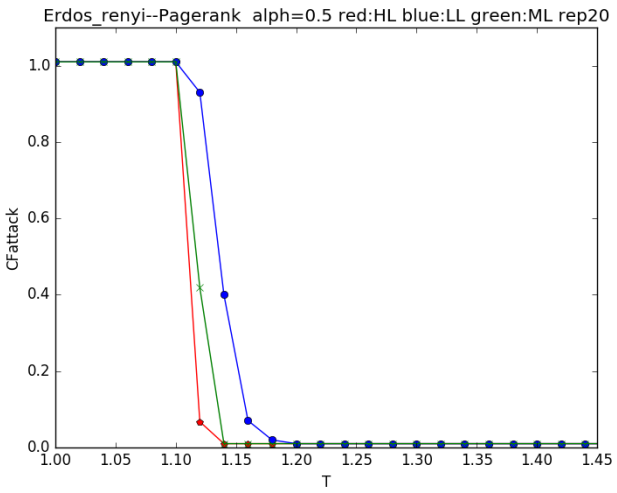

Random_regular--Pagerank alph=0.1 red:HL blue:LL green:ML rep20
CFattack
T

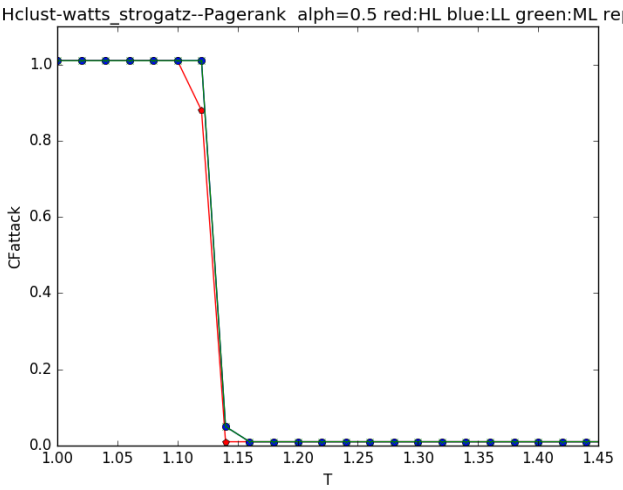
PageRank 0.5

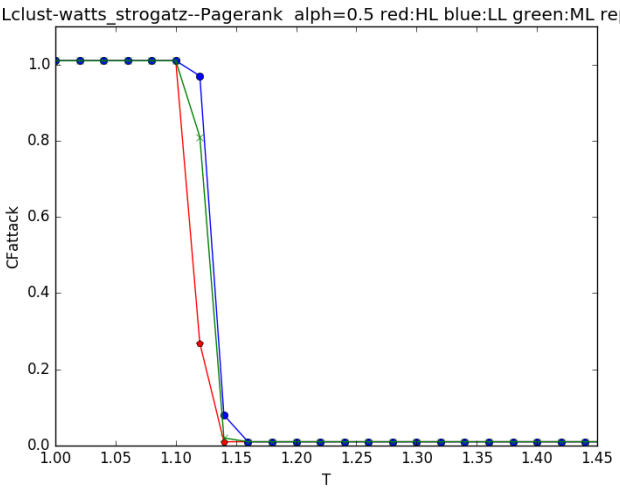

2D-geographical_threshold--Pagerank alph=0.5 red:HL blue:LL green:ML rep20
CFattack
T

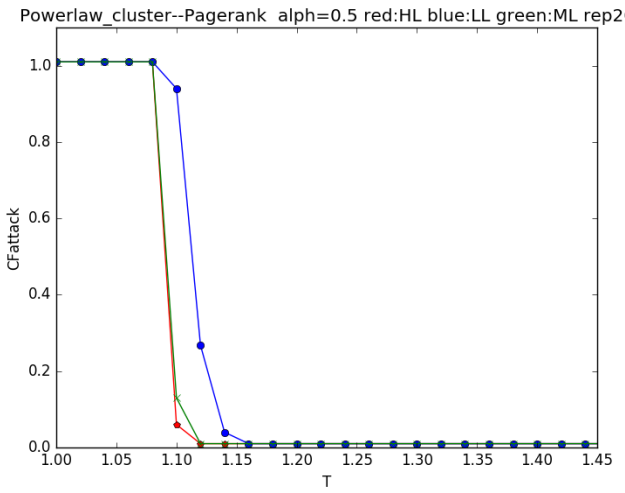

3D-geographical_threshold--Pagerank alph=0.5 red:HL blue:LL green:ML rep20
CFattack
T

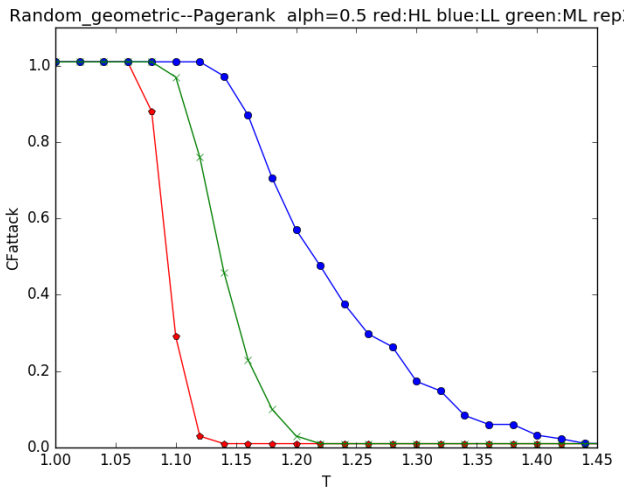

Barabasi_albert--Pagerank alph=0.5 red:HL blue:LL green:ML rep20
CFattack
T

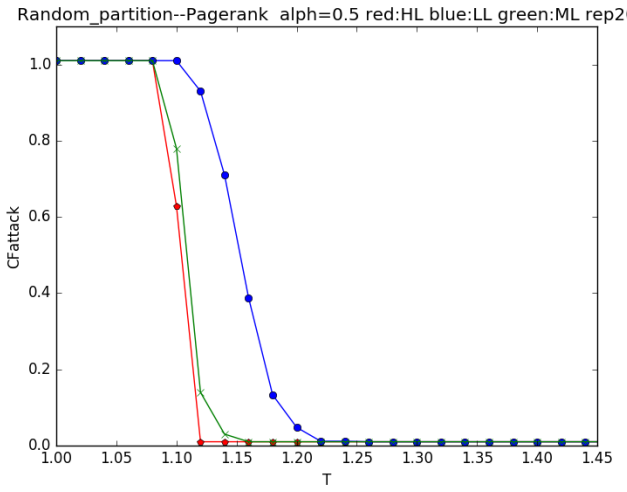

Erdos_renyi--Pagerank alph=0.5 red:HL blue:LL green:ML rep20
CFattack
T

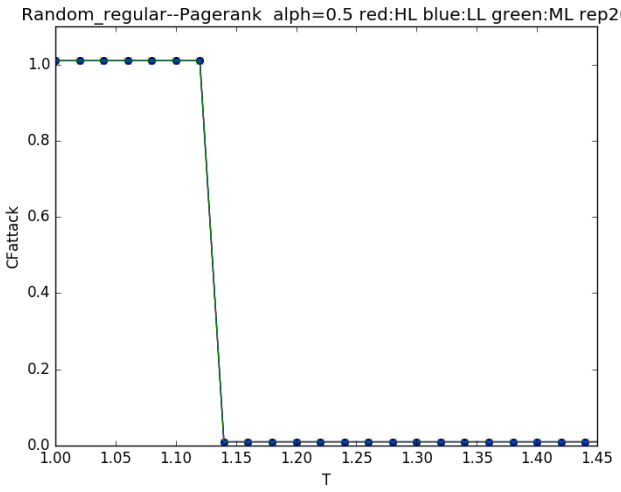

Hclust-watts_strogatz--Pagerank alph=0.5 red:HL blue:LL green:ML rep20
CFattack
T


Lclust-watts_strogatz--Pagerank alph=0.5 red:HL blue:LL green:ML rep20
CFattack
T



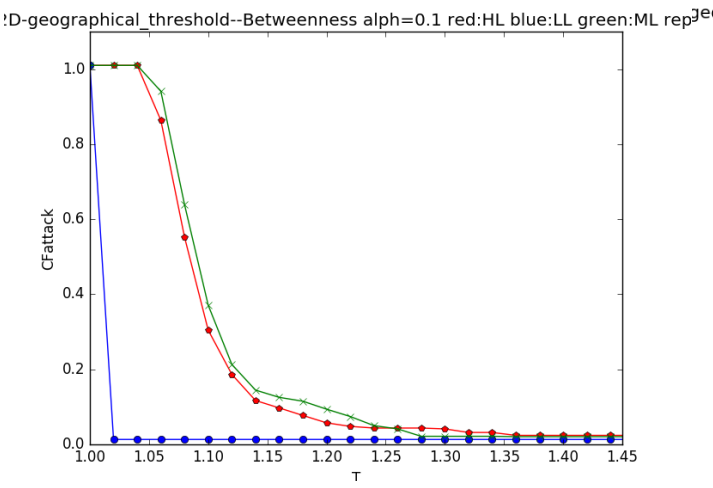


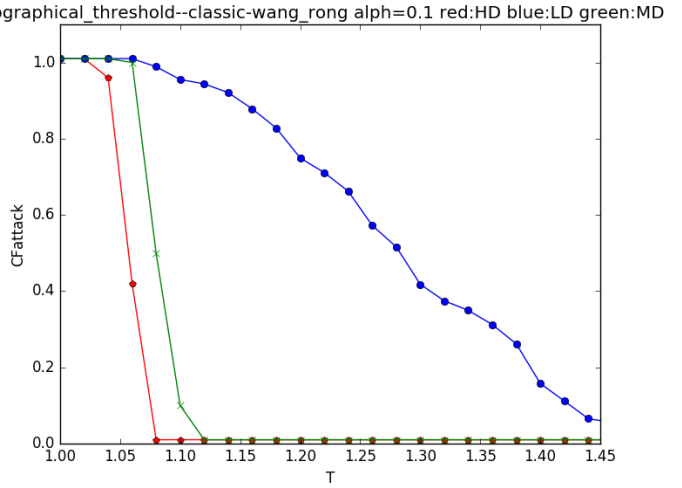


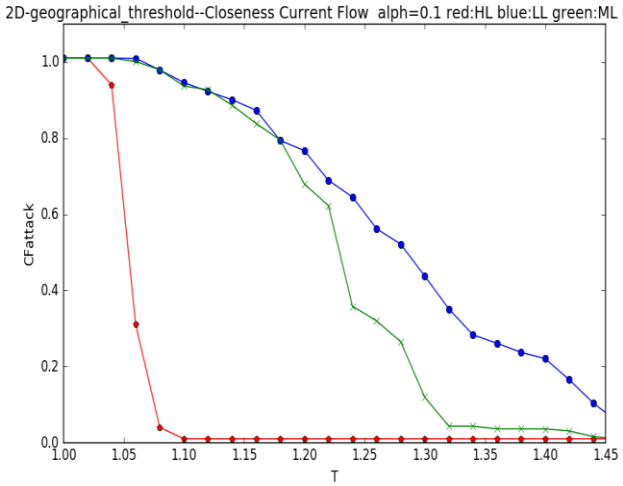


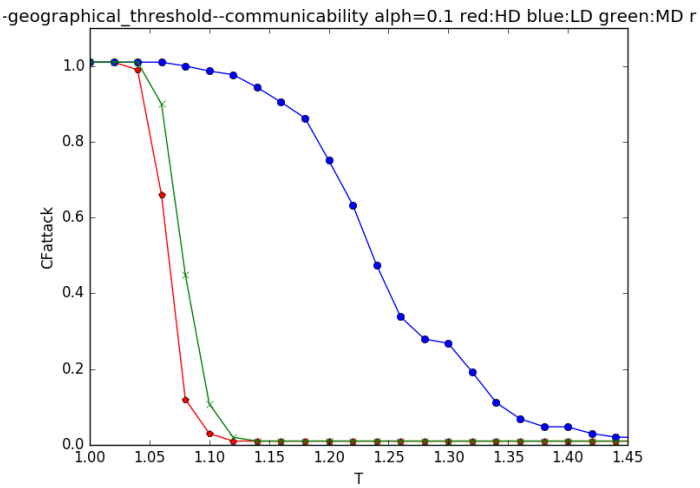
6.3.2 Sorted by Graph family

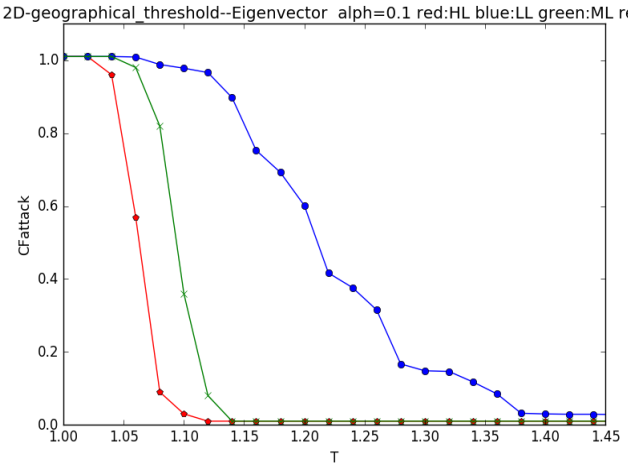
2D Geographical 0.1

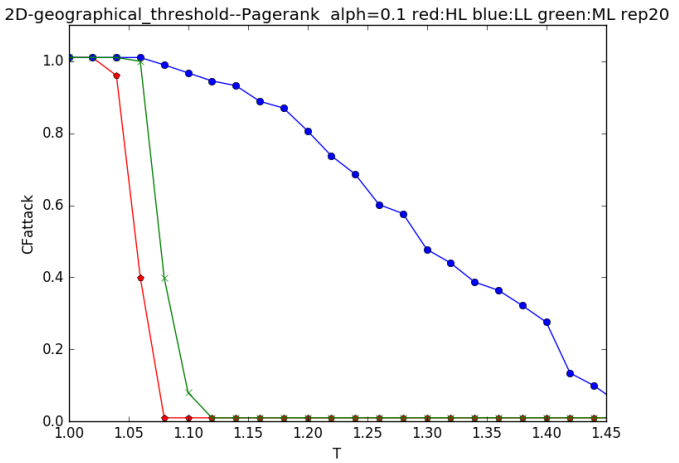




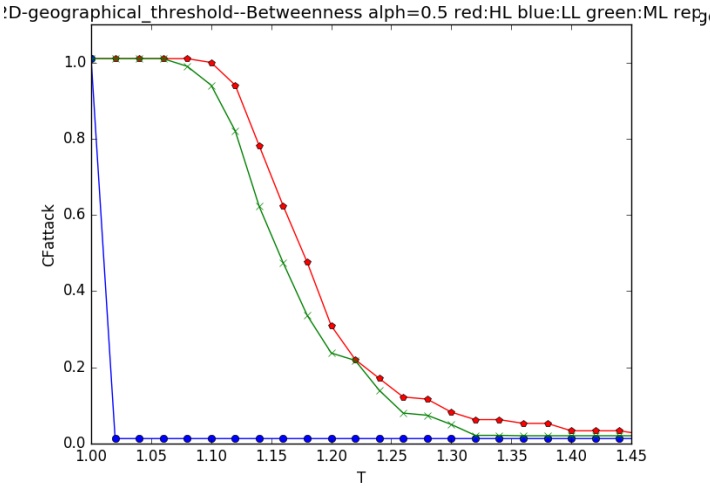


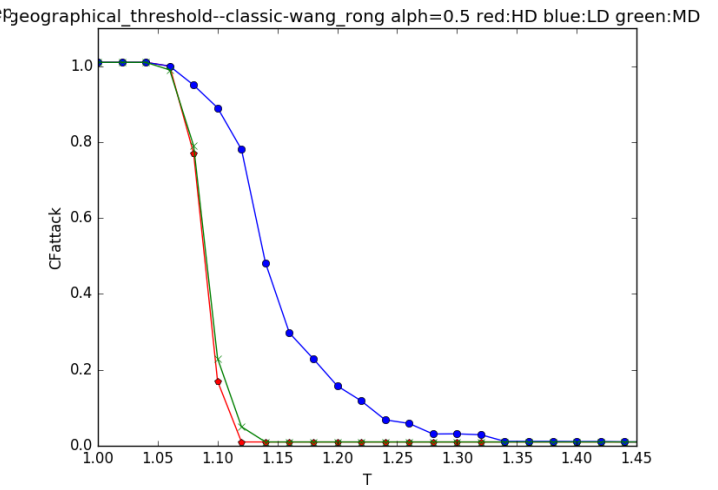


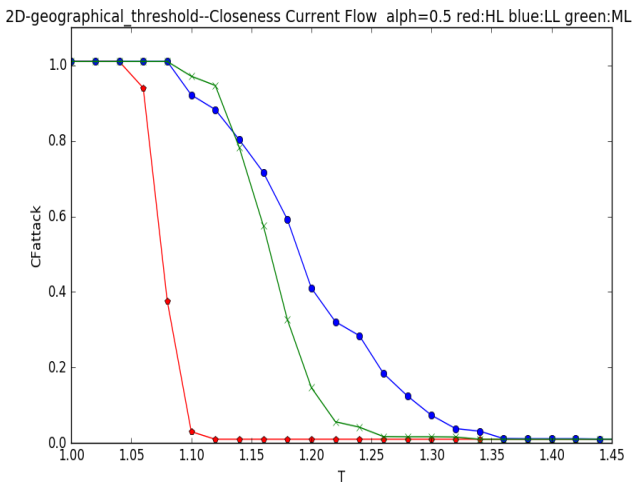


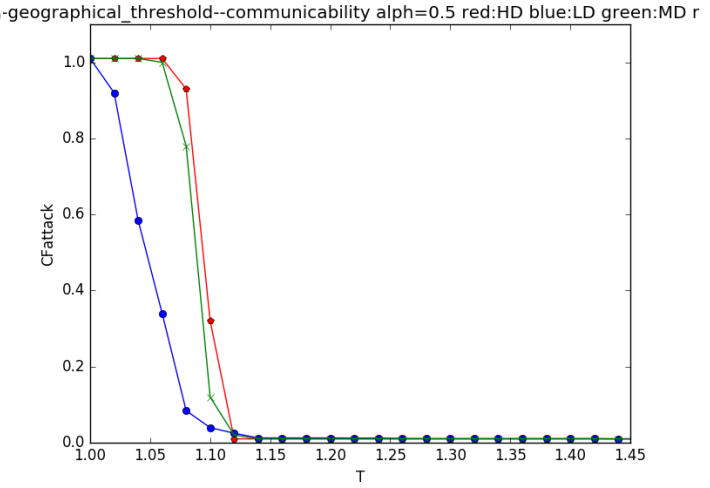


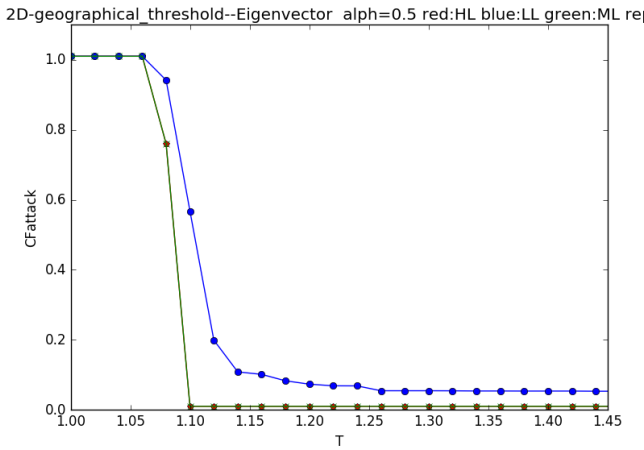
2D Geographical 0.5

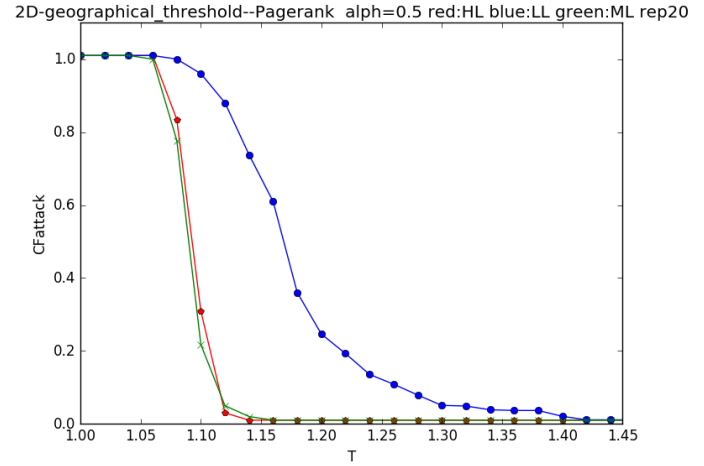






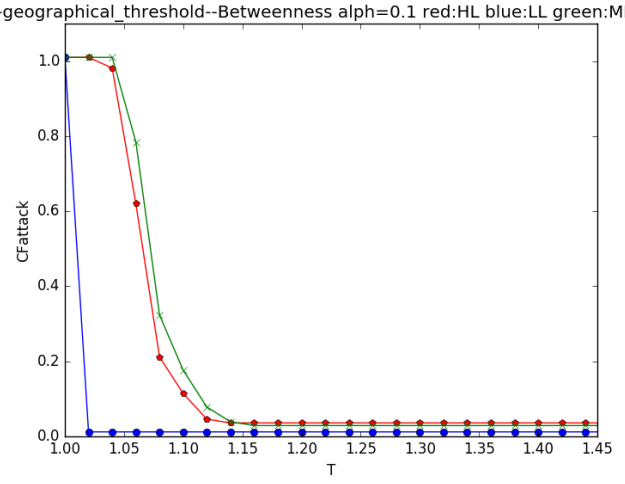




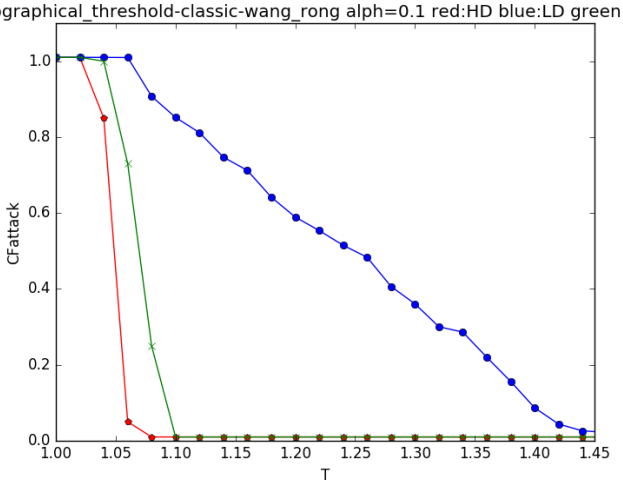


3D Geographical 0.1

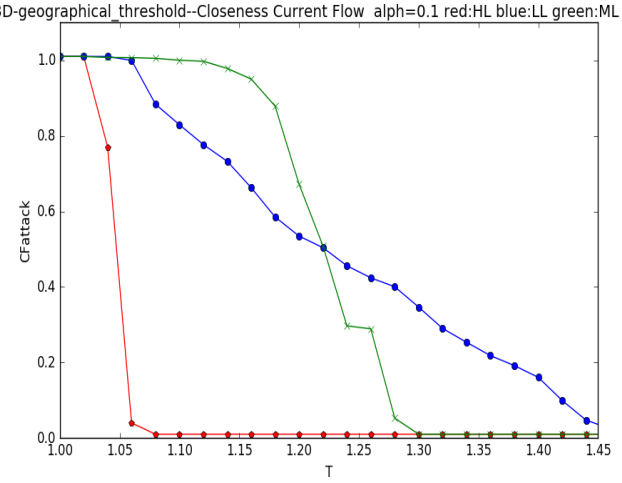
3D-geographical_threshold--Betweenness alph=0.1 red:HL blue:LL green:ML rep



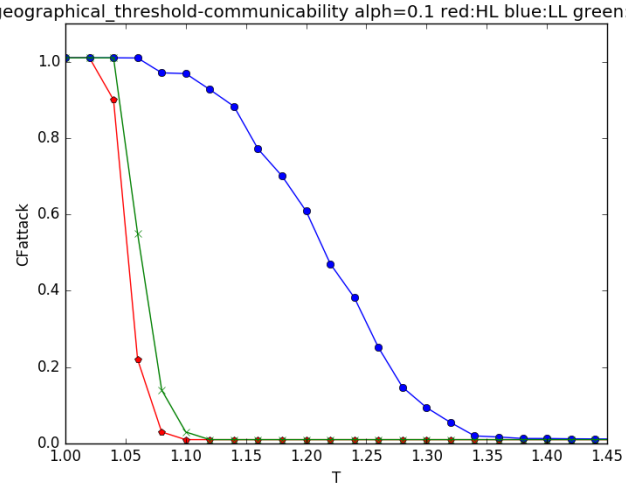
geographical_threshold-classic-wang_rong alph=0.1 red:HD blue:LD green:MD



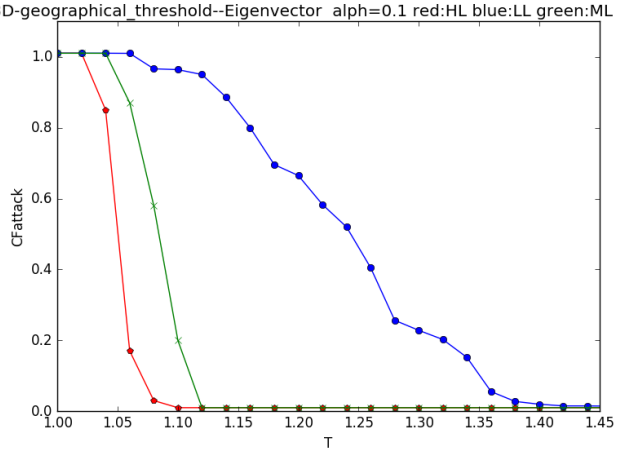
3D-geographical_threshold--Closeness Current Flow alph=0.1 red:HL blue:LL green:ML rep20



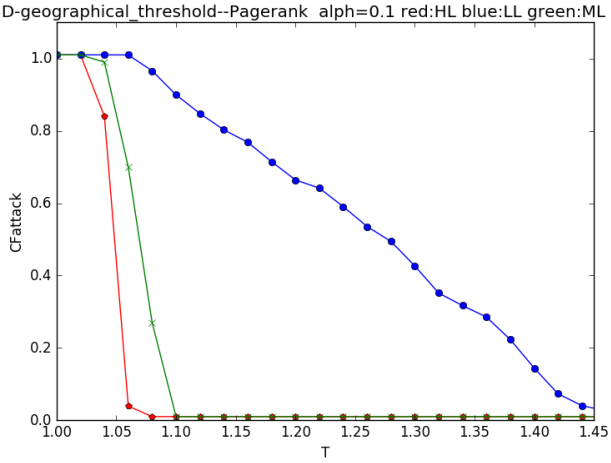
-geographical_threshold-communicability alph=0.1 red:HL blue:LL green:ML re



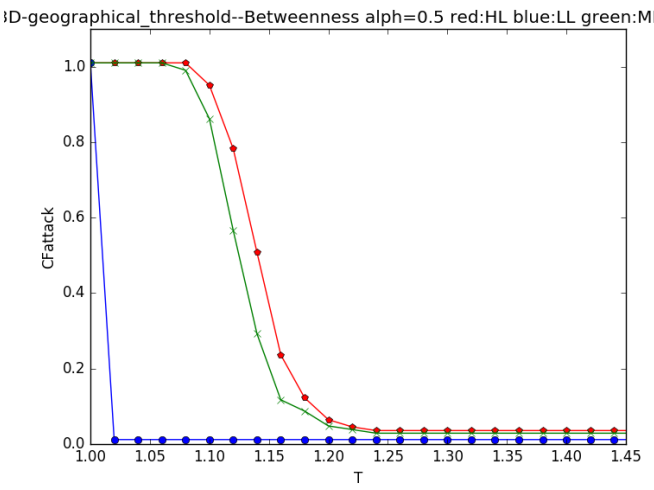
3D-geographical_threshold--Eigenvector alph=0.1 red:HL blue:LL green:ML rep20

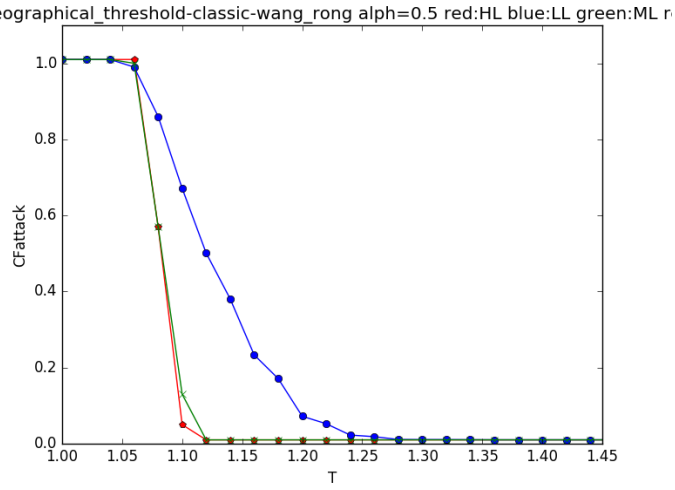


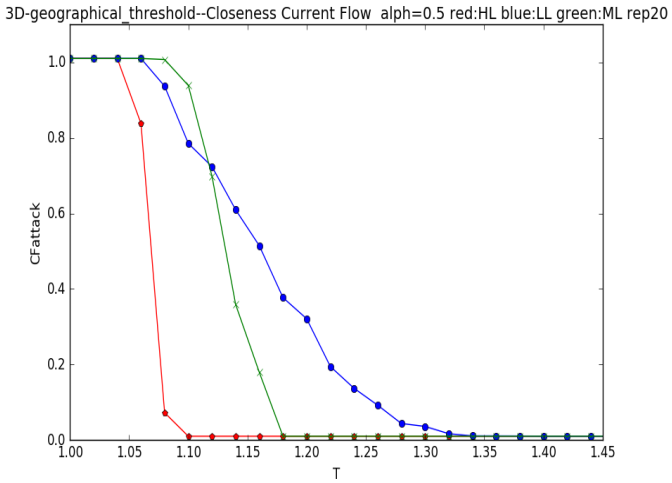
3D-geographical_threshold--Pagerank alph=0.1 red:HL blue:LL green:ML rep20

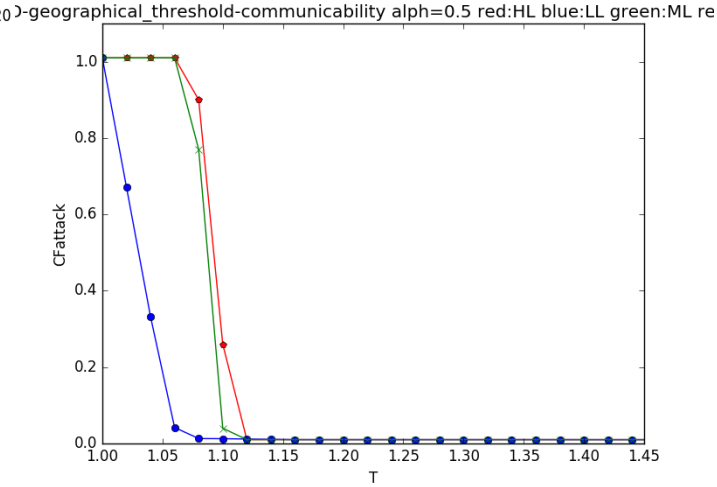


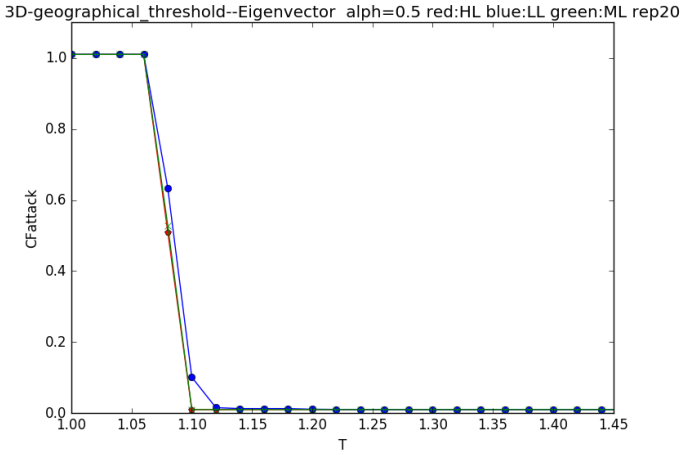
3D Geographical 0.5

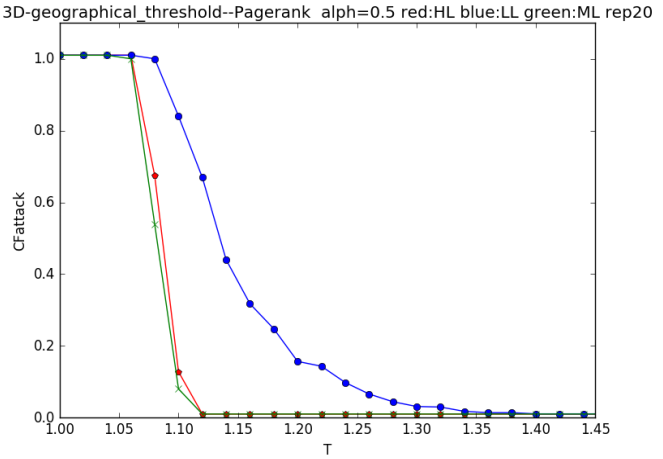




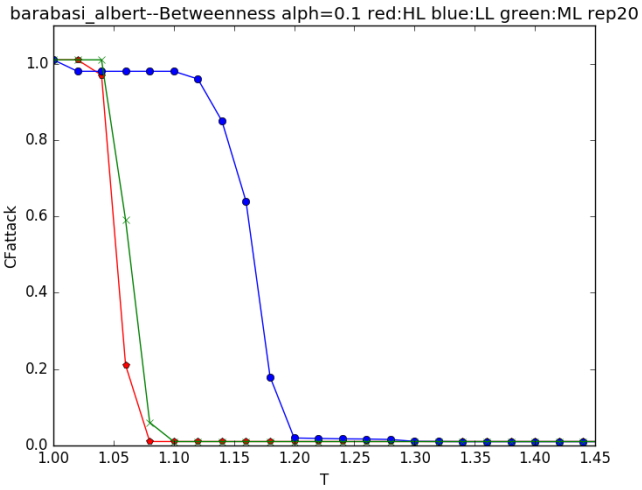


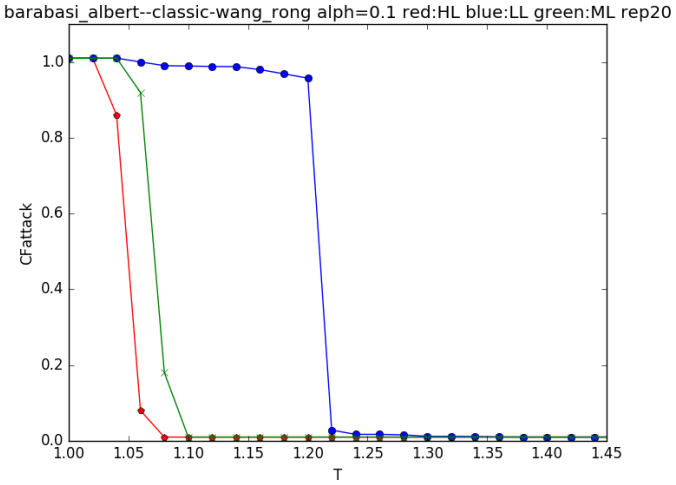


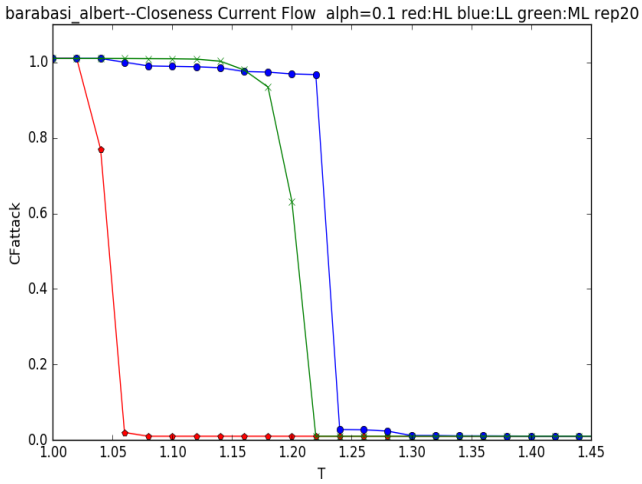


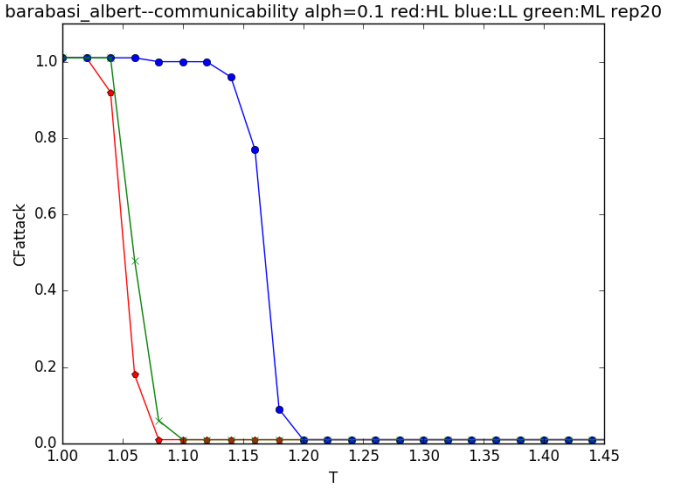


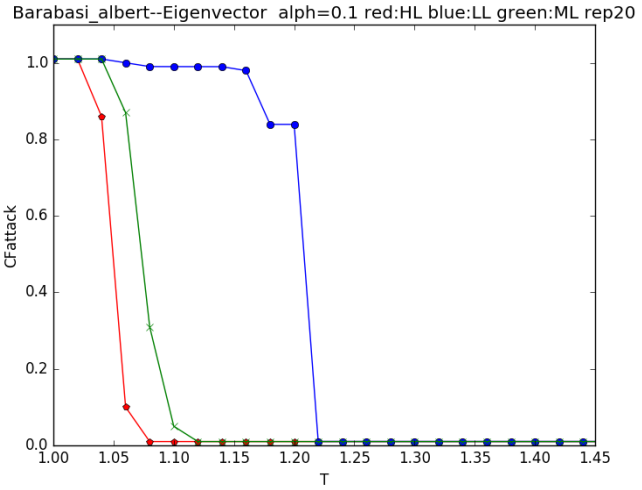
Barabasi 0.1

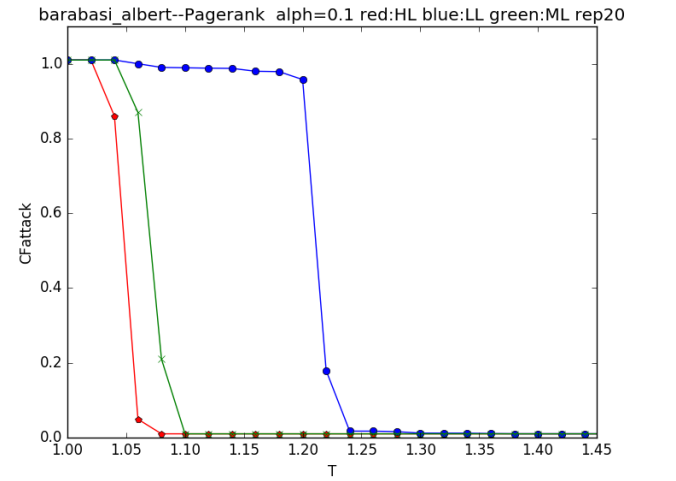

barabasi_albert--Betweenness alph=0.1 red:HL blue:LL green:ML rep20
CFattack
T


barabasi_albert--classic-wang_rong alph=0.1 red:HL blue:LL green:ML rep20
CFattack
T

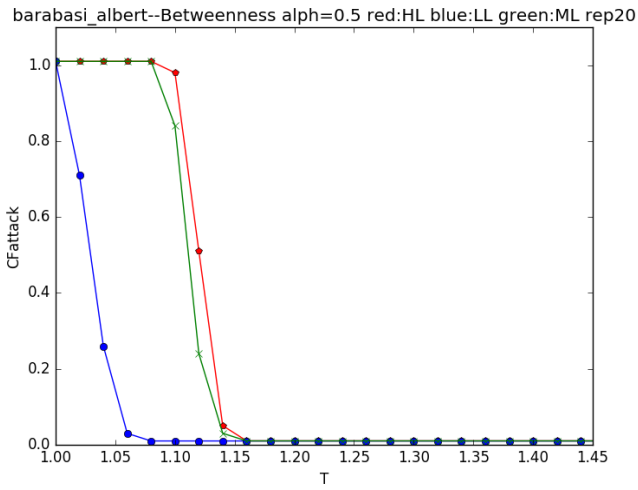

barabasi_albert--Closeness Current Flow alph=0.1 red:HL blue:LL green:ML rep20
CFattack
T

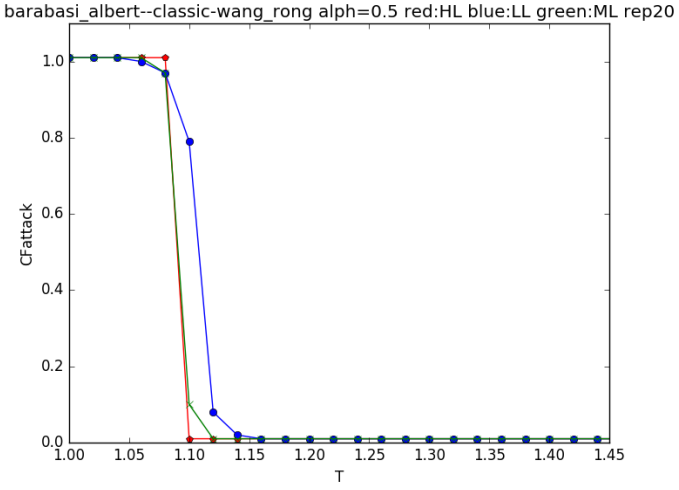

barabasi_albert--communicability alph=0.1 red:HL blue:LL green:ML rep20
CFattack
T

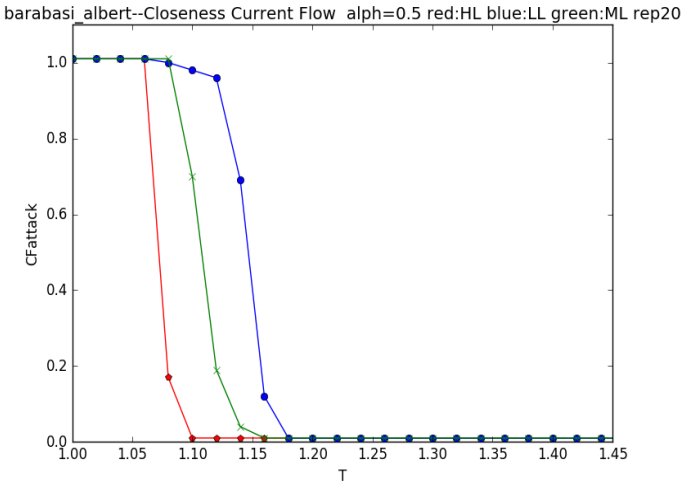

Barabasi_albert--Eigenvector alph=0.1 red:HL blue:LL green:ML rep20
CFattack
T

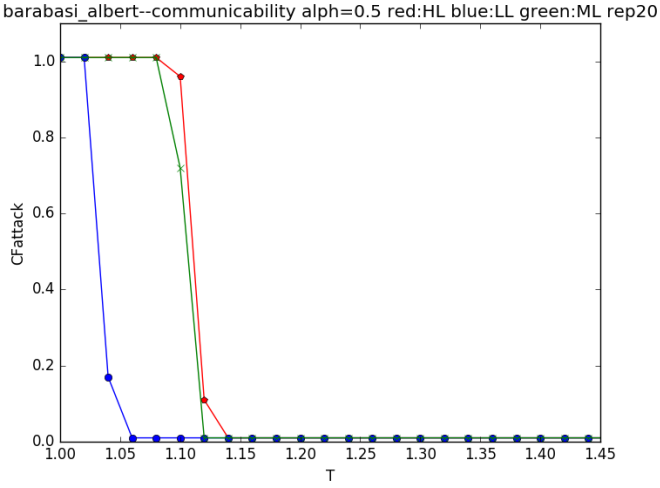

barabasi_albert--Pagerank alph=0.1 red:HL blue:LL green:ML rep20
CFattack
T

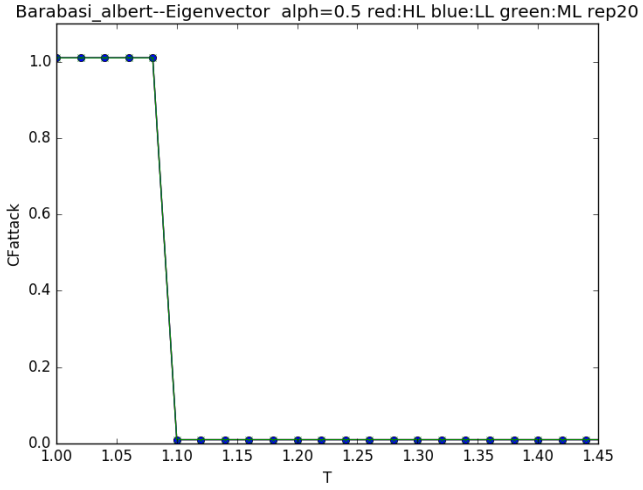
Barabasi 0.5

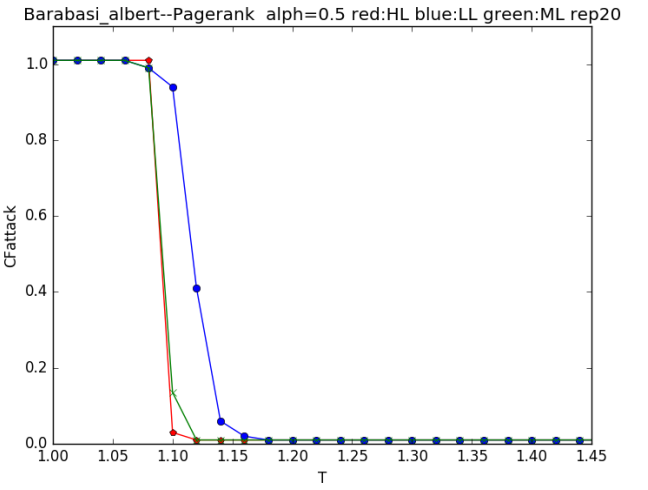

barabasi_albert--Betweenness alph=0.5 red:HL blue:LL green:ML rep20
CFattack
T


barabasi_albert--classic-wang_rong alph=0.5 red:HL blue:LL green:ML rep20
CFattack
T

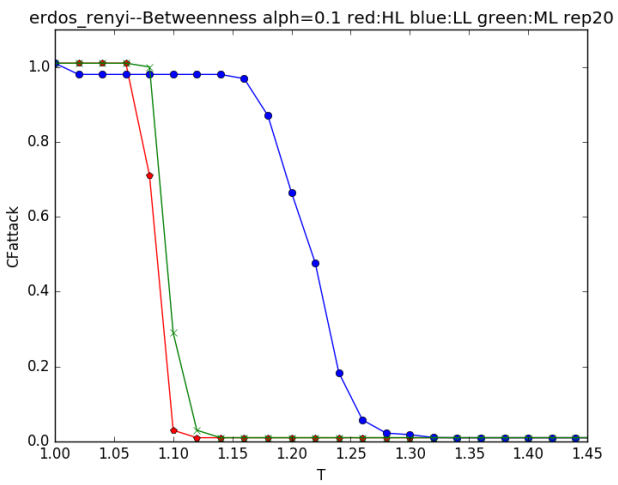

barabasi_albert--Closeness Current Flow alph=0.5 red:HL blue:LL green:ML rep20
CFattack
T

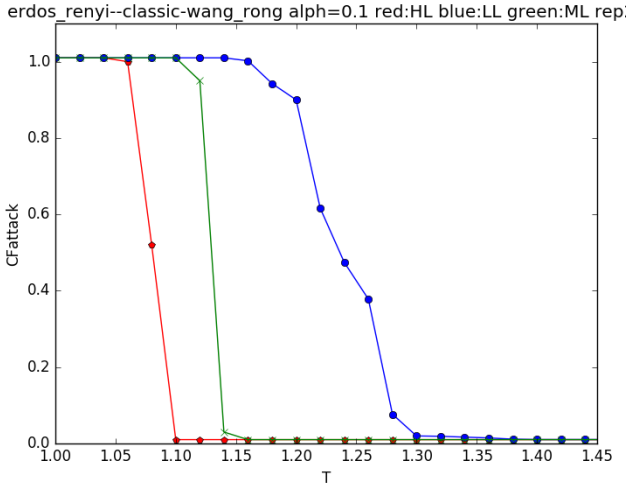

barabasi_albert--communicability alph=0.5 red:HL blue:LL green:ML rep20
CFattack
T

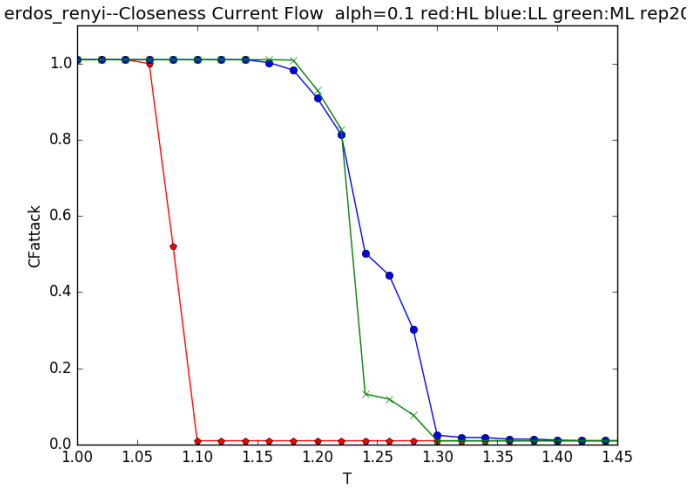

Barabasi_albert--Eigenvector alph=0.5 red:HL blue:LL green:ML rep20
CFattack
T

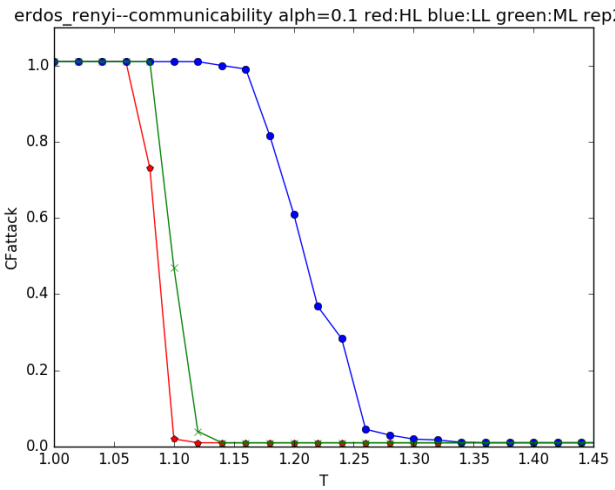

Barabasi_albert--Pagerank alph=0.5 red:HL blue:LL green:ML rep20
CFattack
T

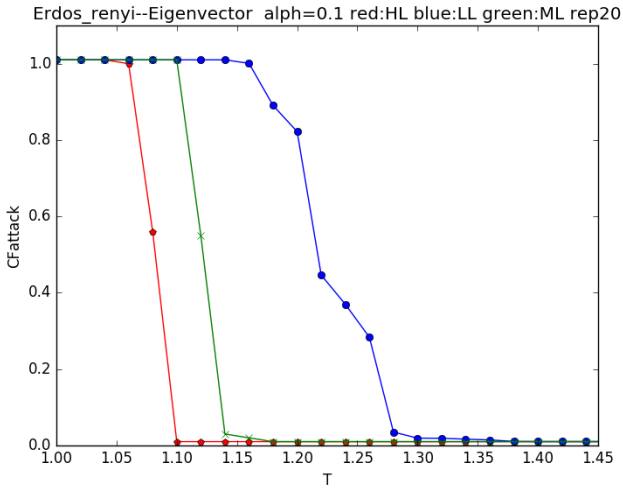
Erdos 0.1

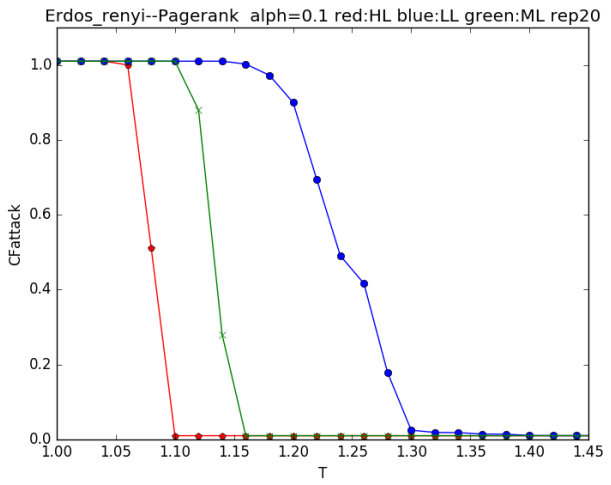

erdos_renyi--Betweenness alph=0.1 red:HL blue:LL green:ML rep20
CFattack
T


erdos_renyi--classic-wang_rong alph=0.1 red:HL blue:LL green:ML rep20
CFattack
T

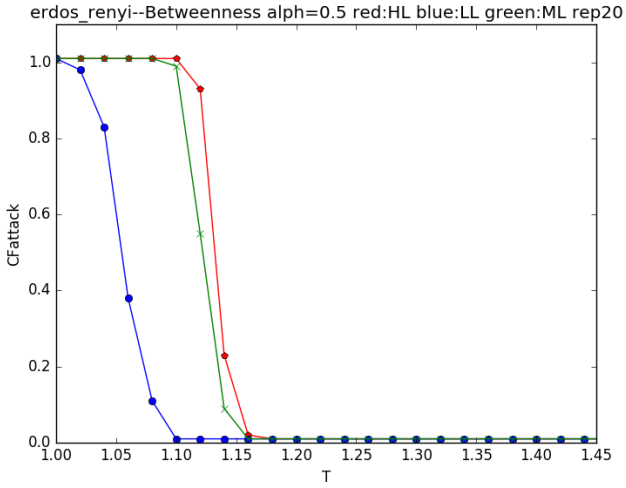

erdos_renyi--Closeness Current Flow alph=0.1 red:HL blue:LL green:ML rep2(
CFattack
T

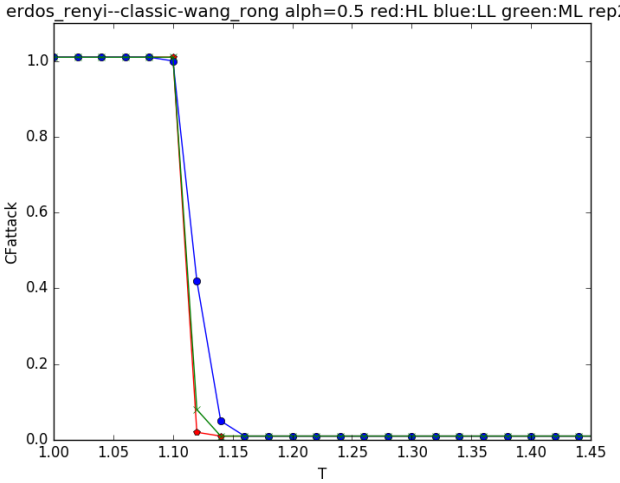

erdos_renyi--communicability alph=0.1 red:HL blue:LL green:ML rep20
CFattack
T

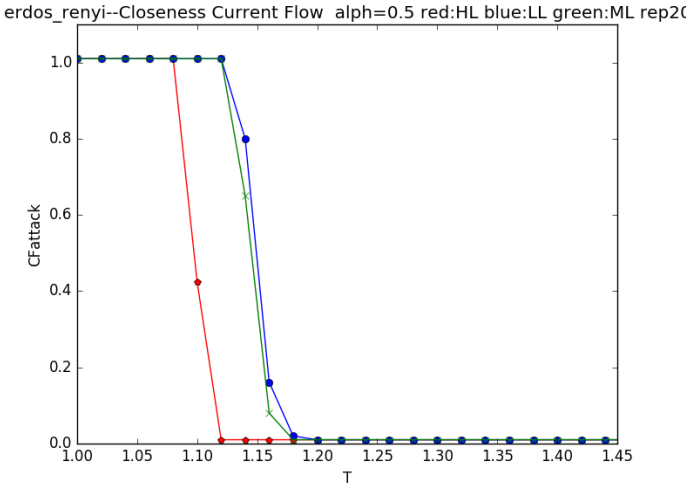

Erdos_renyi--Eigenvector alph=0.1 red:HL blue:LL green:ML rep20
CFattack
T

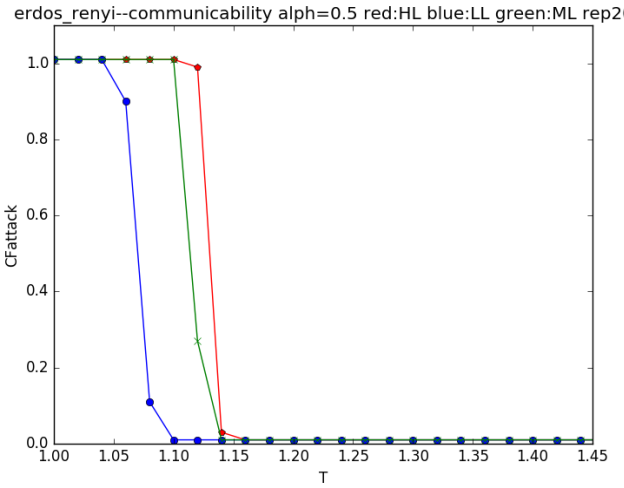

Erdos_renyi--Pagerank alph=0.1 red:HL blue:LL green:ML rep20
CFattack
T

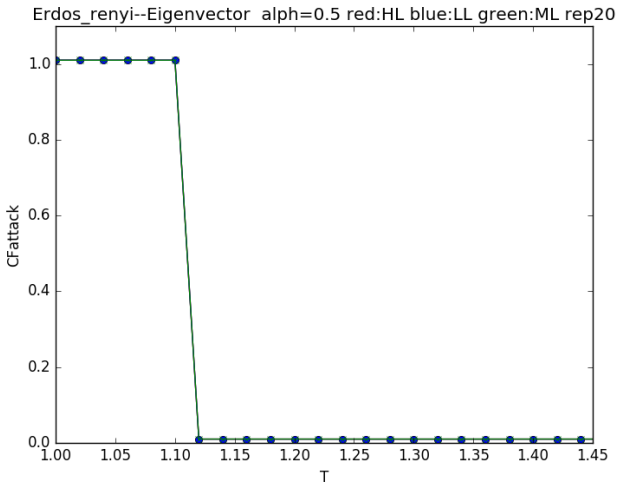
Erdos 0.5

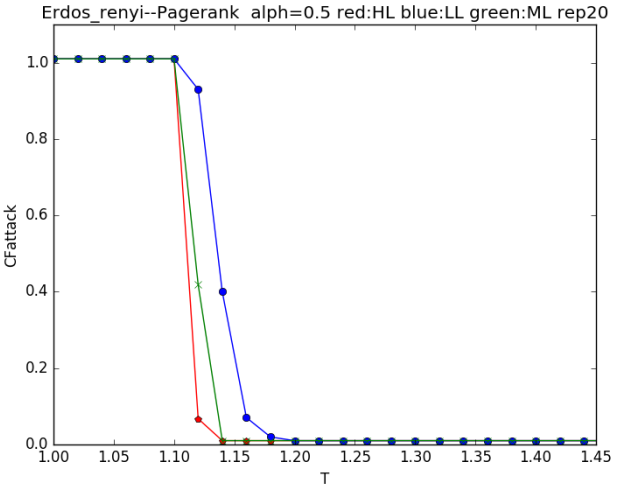

erdos_renyi--Betweenness alph=0.5 red:HL blue:LL green:ML rep20
CFattack
T


erdos_renyi--classic-wang_rong alph=0.5 red:HL blue:LL green:ML rep20
CFattack
T


erdos_renyi--Closeness Current Flow alph=0.5 red:HL blue:LL green:ML rep2(
CFattack
T

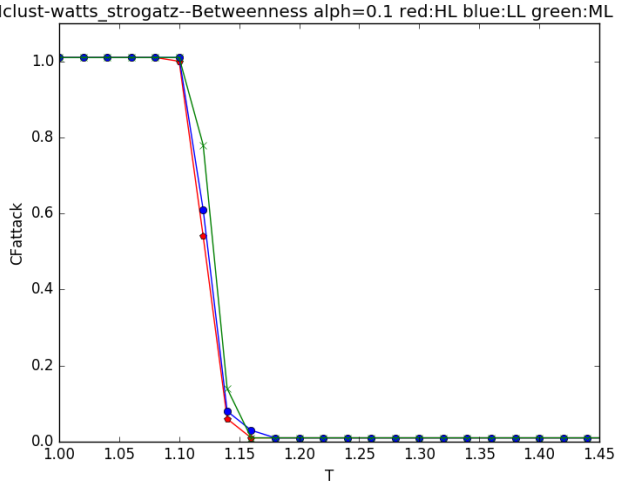

erdos_renyi--communicability alph=0.5 red:HL blue:LL green:ML rep20
CFattack
T


Erdos_renyi--Eigenvector alph=0.5 red:HL blue:LL green:ML rep20
CFattack
T

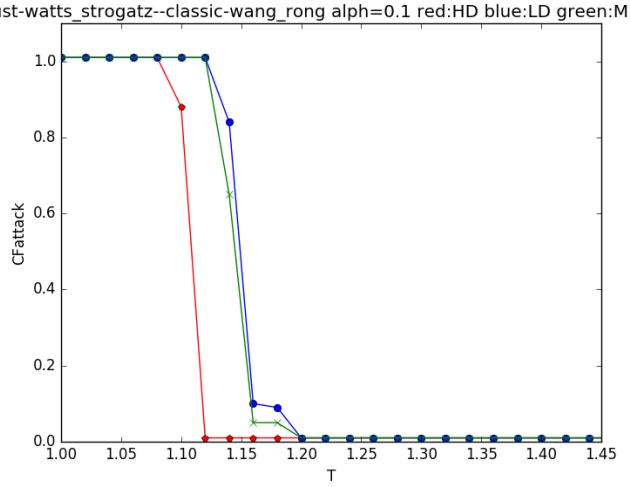

Erdos_renyi--Pagerank alph=0.5 red:HL blue:LL green:ML rep20
CFattack
T

Watts Hclust 0.1

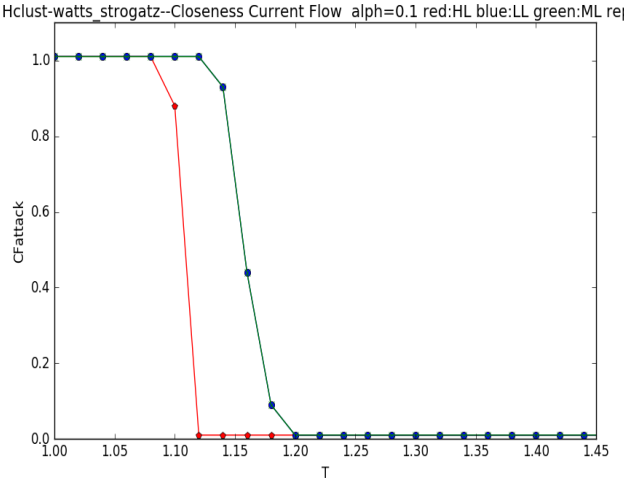
Hclust-watts_strogatz--Betweenness alph=0.1 red:HL blue:LL green:ML rep20



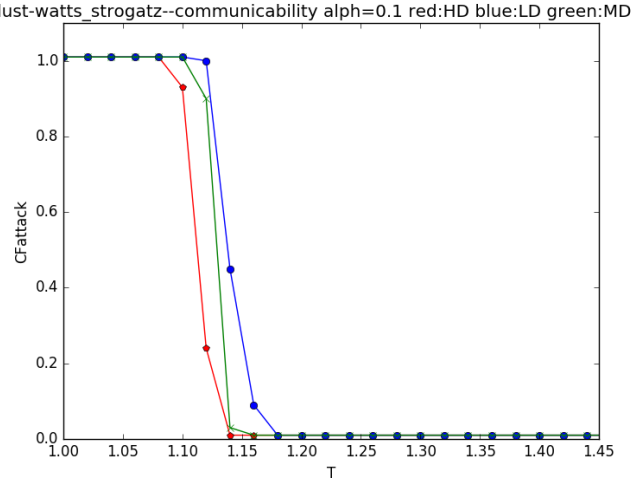
Hclust-watts_strogatz--classic-wang_rong alph=0.1 red:HD blue:LD green:MD rep



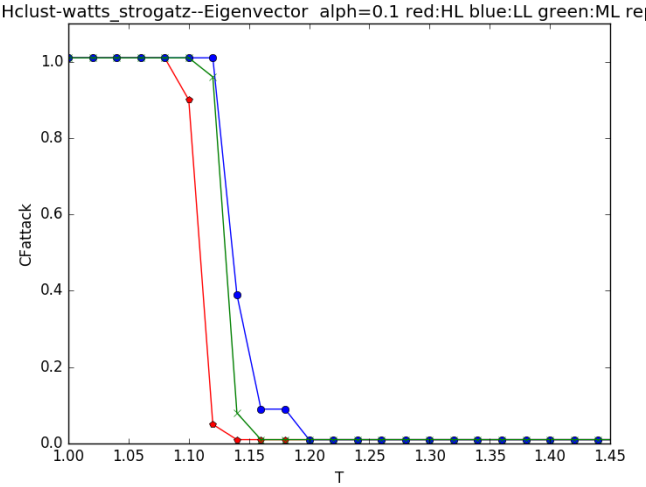
Hclust-watts_strogatz--Closeness Current Flow alph=0.1 red:HL blue:LL green:ML rep20



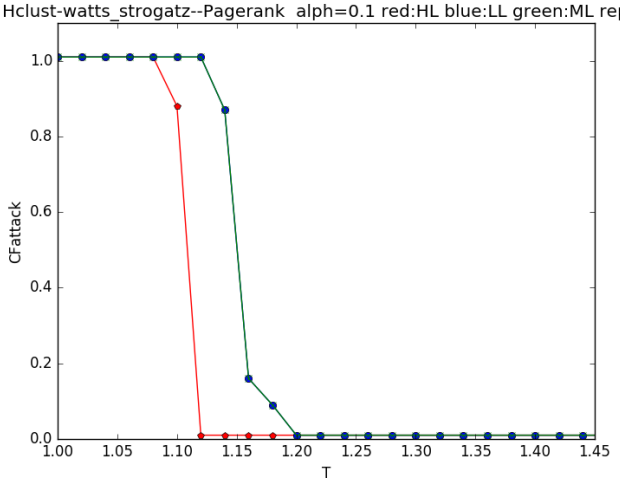
Hclust-watts_strogatz--communicability alph=0.1 red:HD blue:LD green:MD rep



Hclust-watts_strogatz--Eigenvector alph=0.1 red:HL blue:LL green:ML rep20

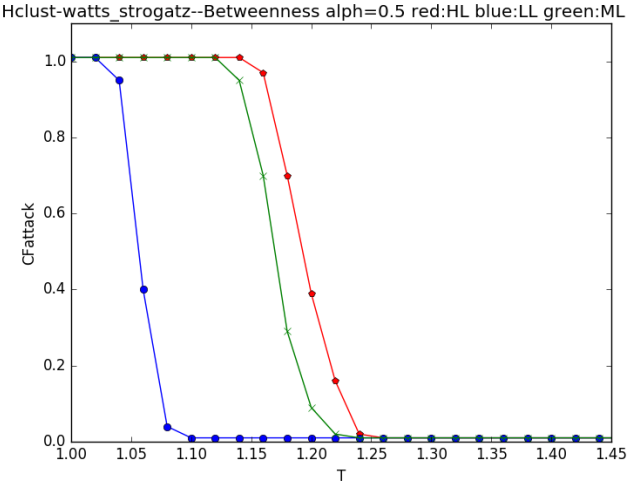


Hclust-watts_strogatz--Pagerank alph=0.1 red:HL blue:LL green:ML rep20

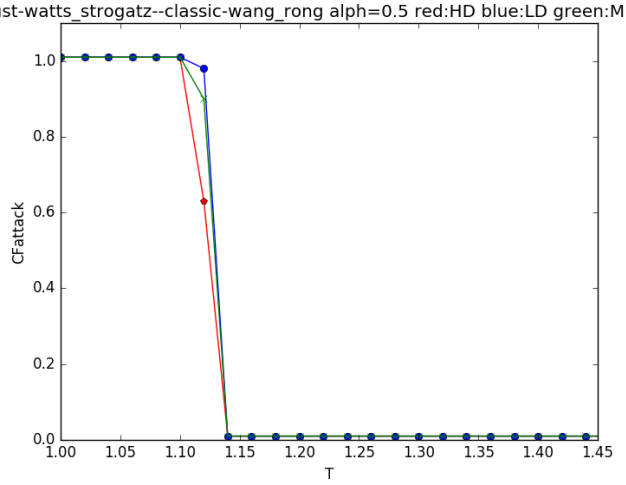


Watts Hclust 0.5

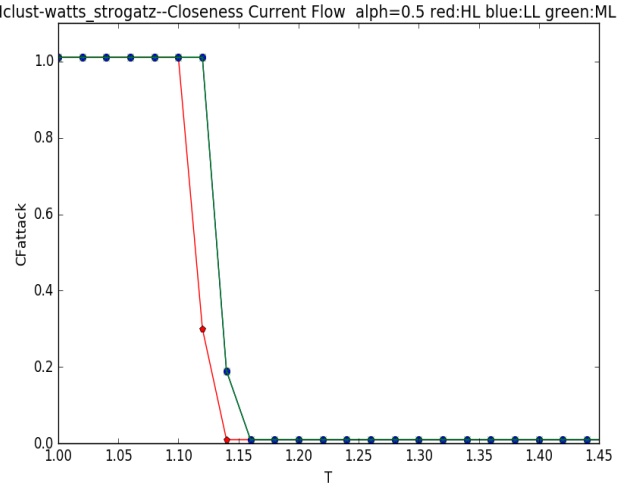
Hclust-watts_strogatz--Betweenness alph=0.5 red:HL blue:LL green:ML rep20



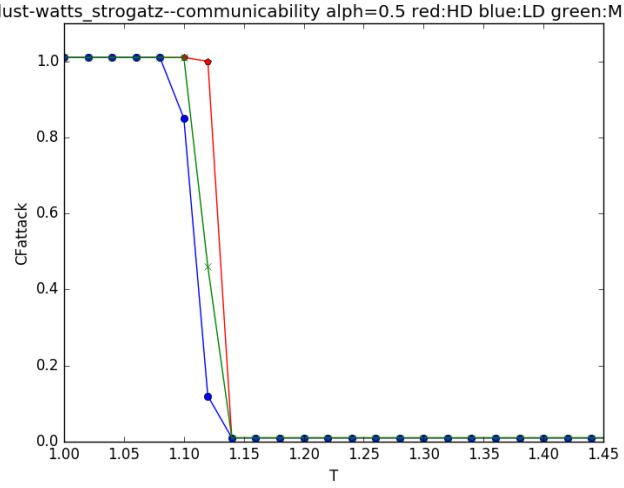
Hclust-watts_strogatz--classic-wang_rong alph=0.5 red:HD blue:LD green:MD rep



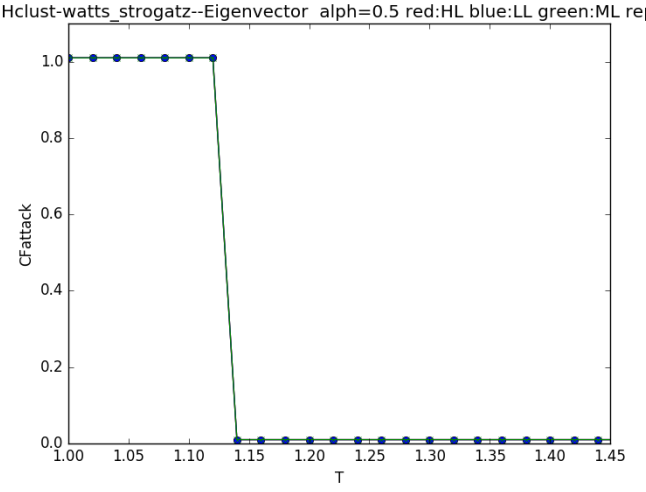
Hclust-watts_strogatz--Closeness Current Flow alph=0.5 red:HL blue:LL green:ML rep20



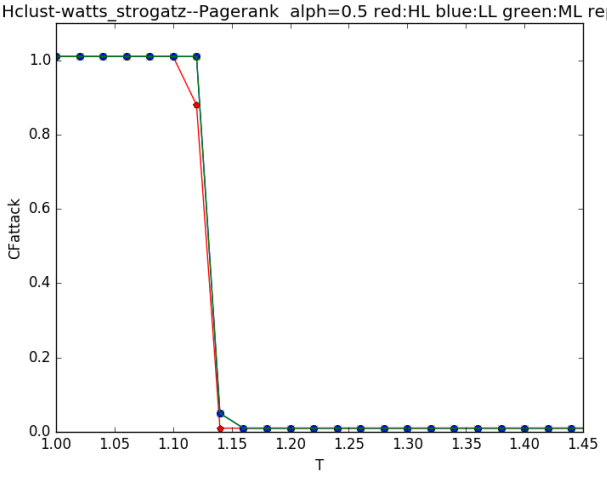
Hclust-watts_strogatz--communicability alph=0.5 red:HD blue:LD green:MD rep



Hclust-watts_strogatz--Eigenvector alph=0.5 red:HL blue:LL green:ML rep20

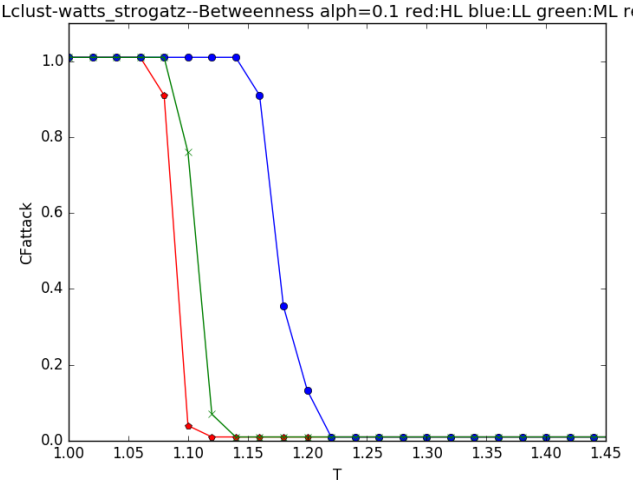


Hclust-watts_strogatz--Pagerank alph=0.5 red:HL blue:LL green:ML rep20

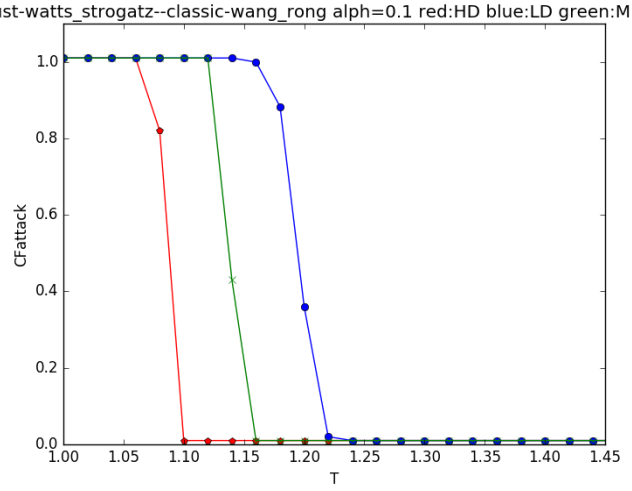


Watts Lclust 0.1

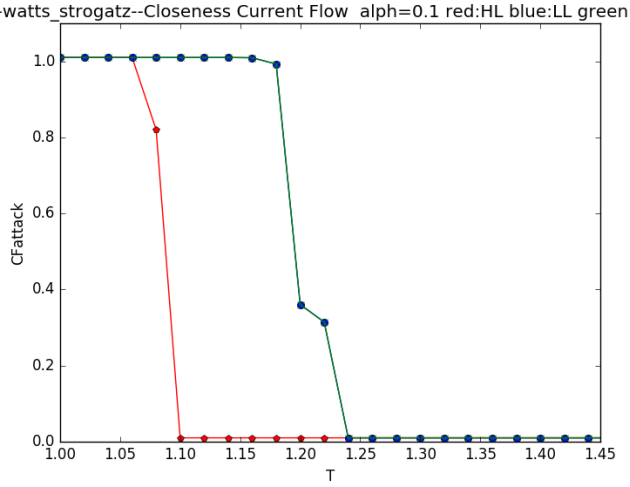
Lclust-watts_strogatz--Betweenness alph=0.1 red:HL blue:LL green:ML rep20



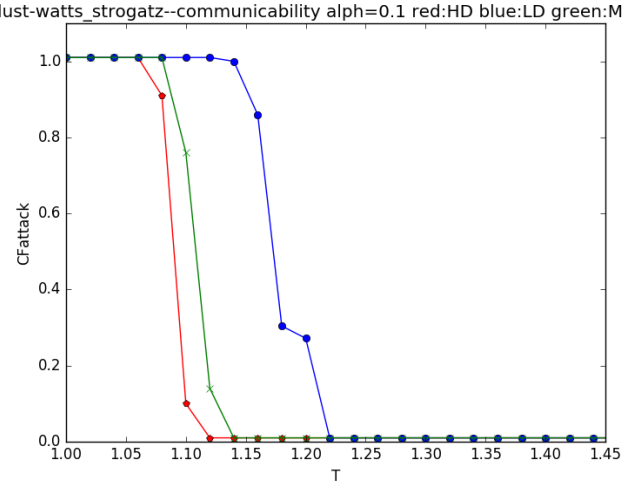
clust-watts_strogatz--classic-wang_rong alph=0.1 red:HD blue:LD green:MD rep



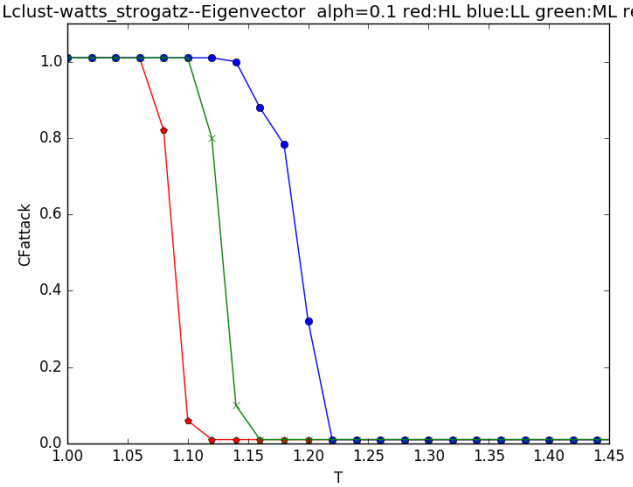
ust-watts_strogatz--Closeness Current Flow alph=0.1 red:HL blue:LL green:ML r



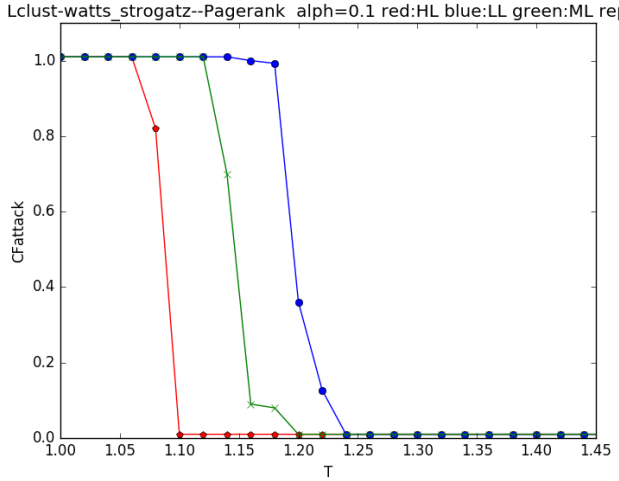
Lclust-watts_strogatz--communicability alph=0.1 red:HD blue:LD green:MD rep2



Lclust-watts_strogatz--Eigenvector alph=0.1 red:HL blue:LL green:ML rep20

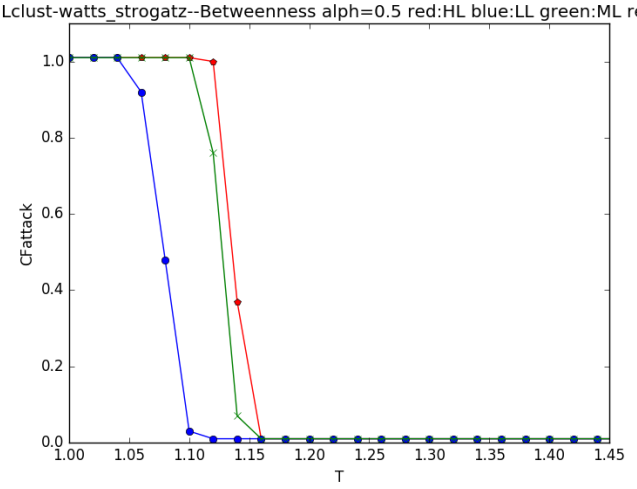


Lclust-watts_strogatz--Pagerank alph=0.1 red:HL blue:LL green:ML rep20

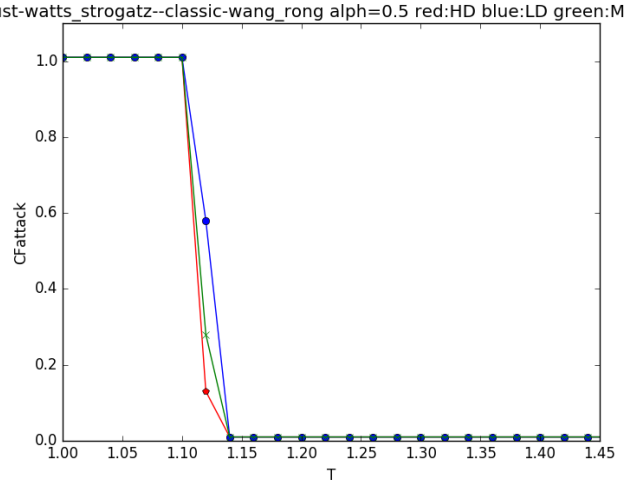


Watts Lcluster 0.5

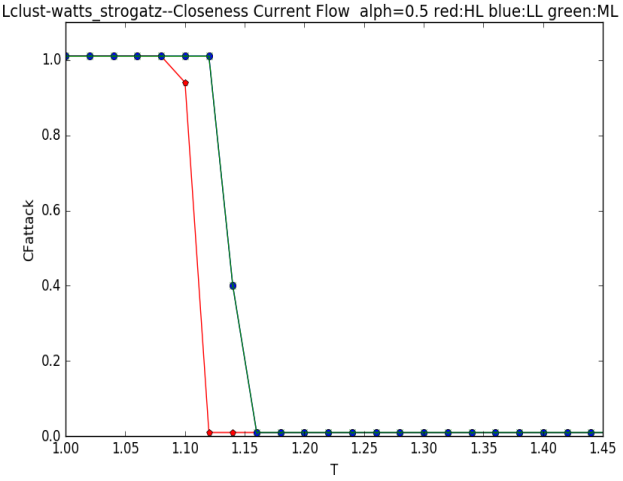
Lclust-watts_strogatz--Betweenness alph=0.5 red:HL blue:LL green:ML rep20



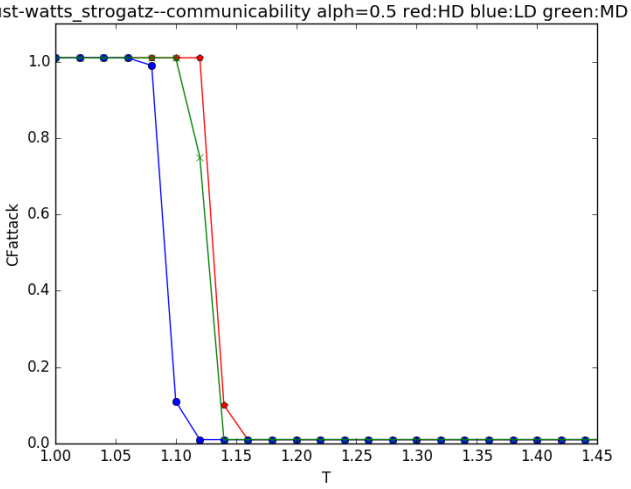
clust-watts_strogatz--classic-wang_rong alph=0.5 red:HD blue:LD green:MD rep



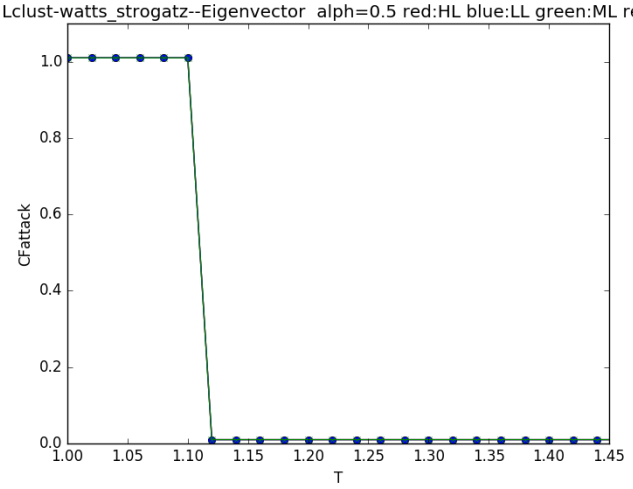
Lclust-watts_strogatz--Closeness Current Flow alph=0.5 red:HL blue:LL green:ML rep20



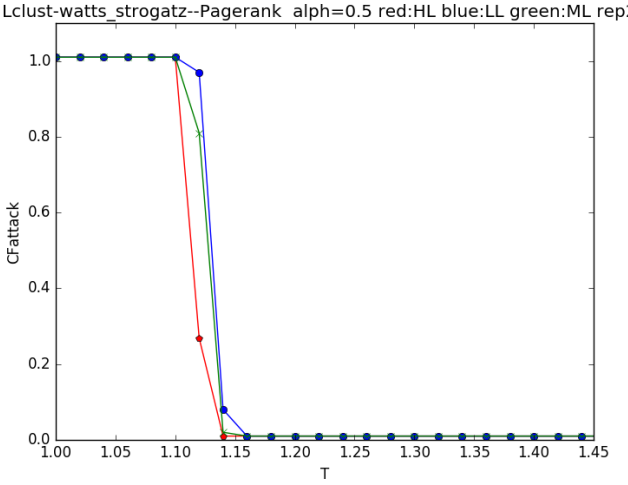
Lclust-watts_strogatz--communicability alph=0.5 red:HD blue:LD green:MD rep2



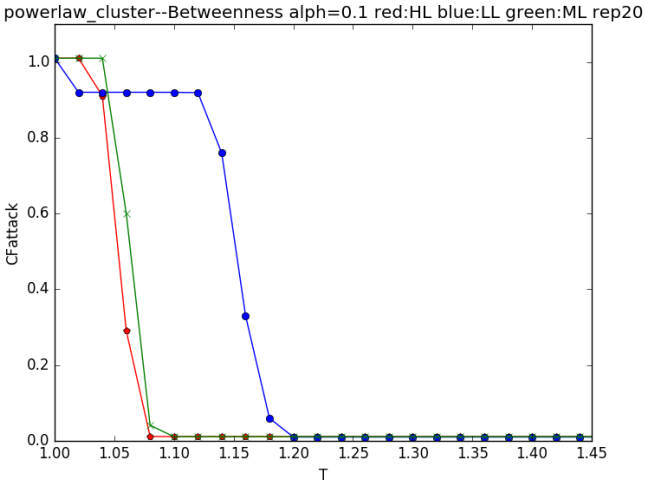
Lclust-watts_strogatz--Eigenvector alph=0.5 red:HL blue:LL green:ML rep20

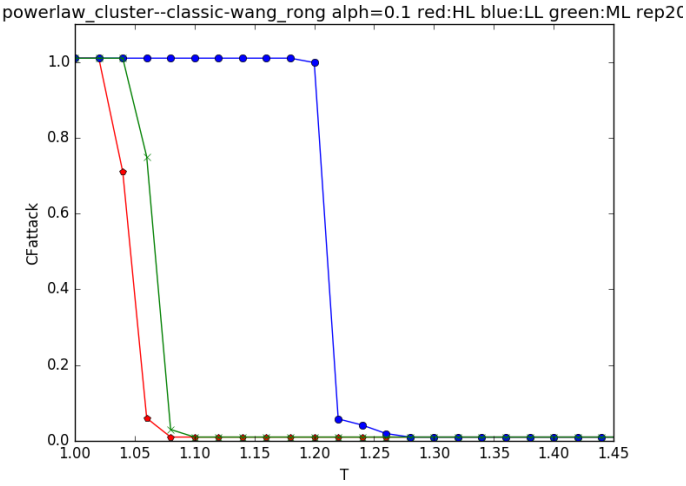


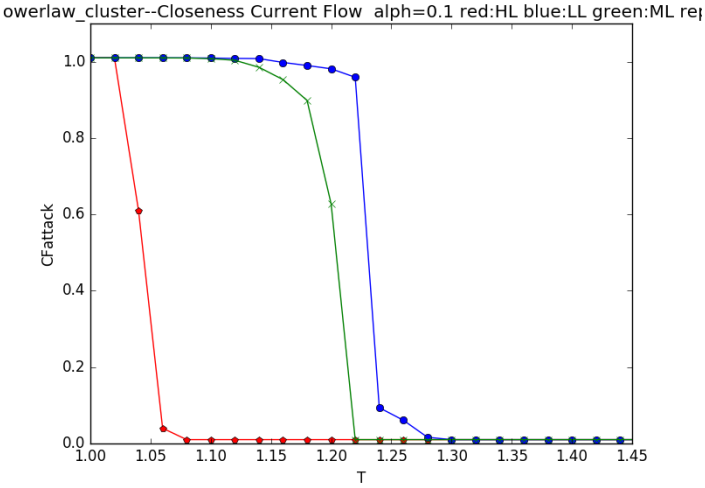
Lclust-watts_strogatz--Pagerank alph=0.5 red:HL blue:LL green:ML rep20

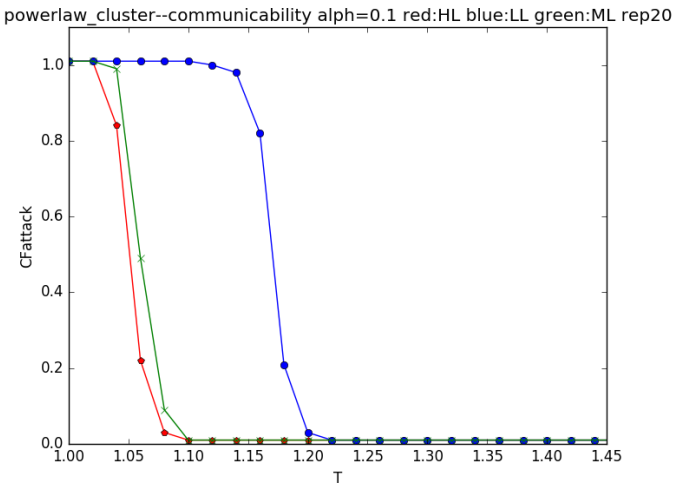


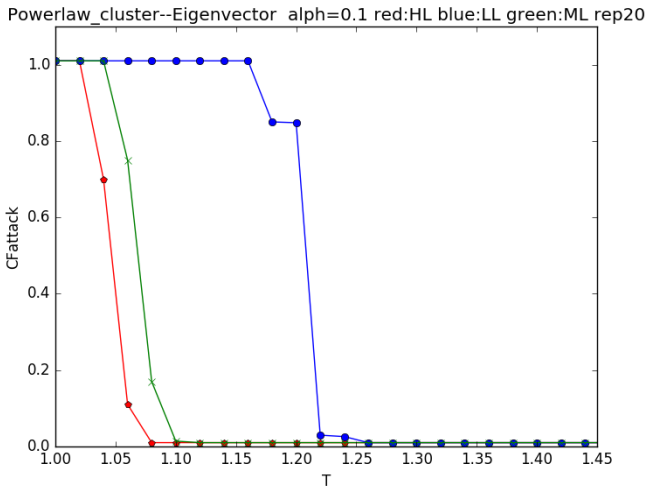
Powerlaw 0.1

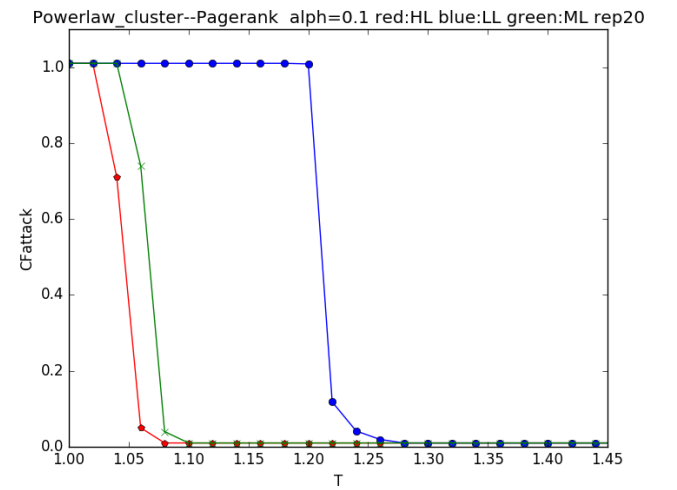

powerlaw_cluster--Betweenness alph=0.1 red:HL blue:LL green:ML rep20
CFattack
T


powerlaw_cluster--classic-wang_rong alph=0.1 red:HL blue:LL green:ML rep20
CFattack
T

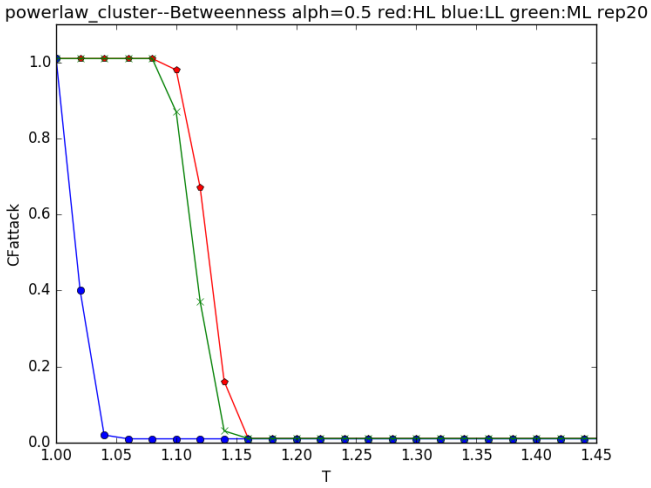

owerlaw_cluster--Closeness Current Flow alph=0.1 red:HL blue:LL green:ML rep
CFattack
T

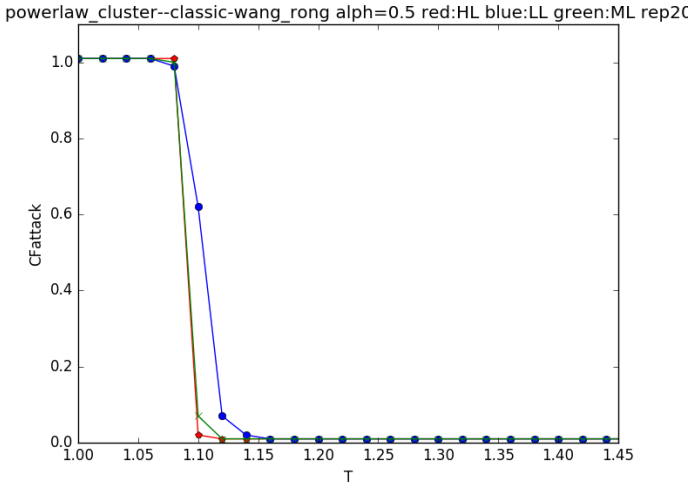

powerlaw_cluster--communicability alph=0.1 red:HL blue:LL green:ML rep20
CFattack
T

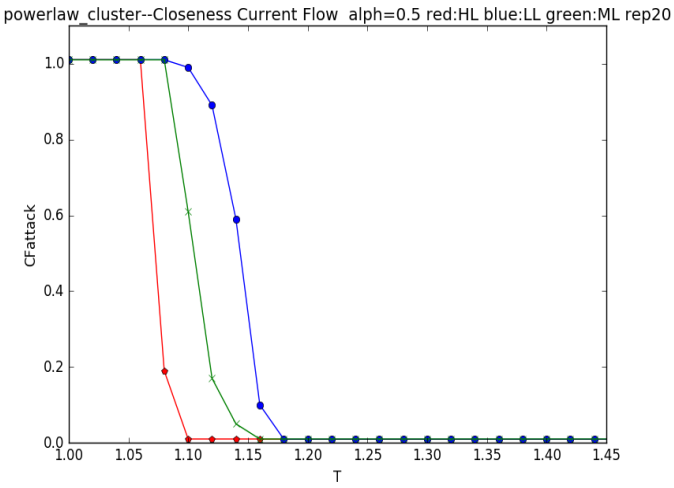

Powerlaw_cluster--Eigenvector alph=0.1 red:HL blue:LL green:ML rep20
CFattack
T

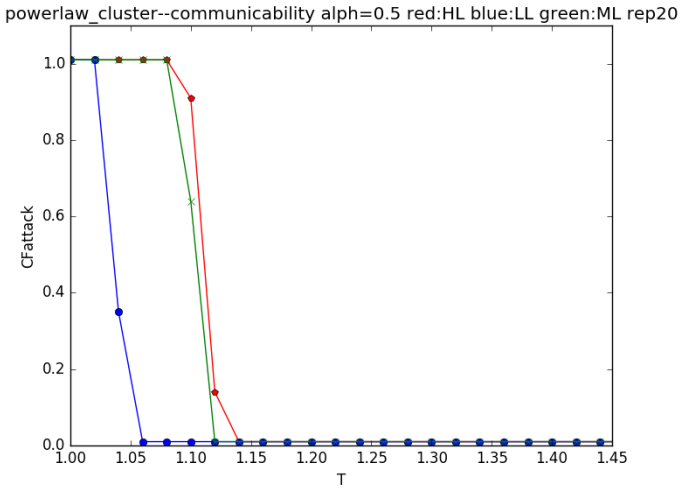

Powerlaw_cluster--Pagerank alph=0.1 red:HL blue:LL green:ML rep20
CFattack
T

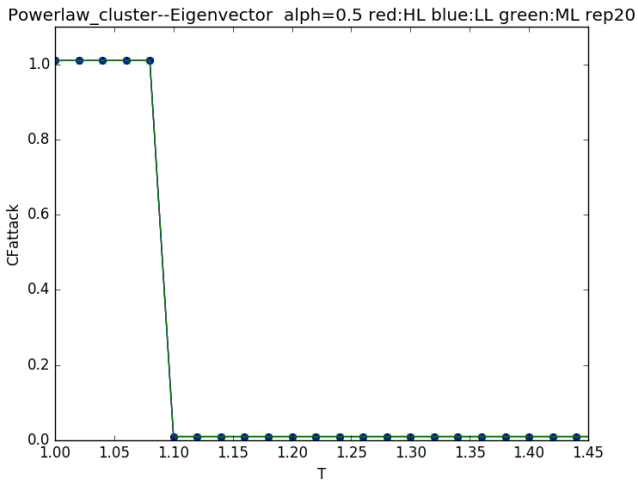
Powerlaw 0.5

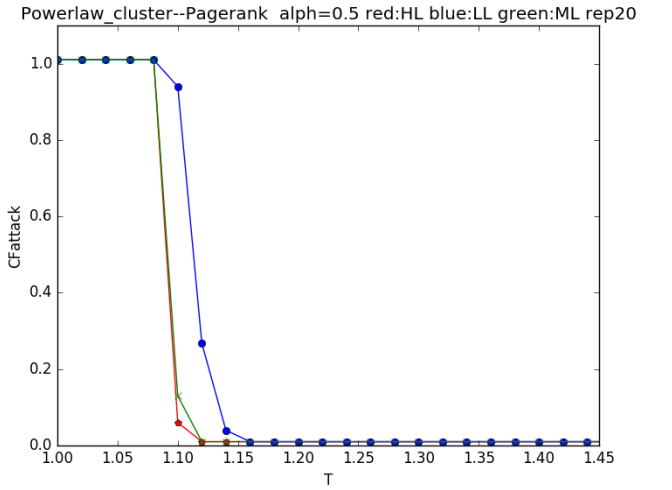

powerlaw_cluster--Betweenness alph=0.5 red:HL blue:LL green:ML rep20
CFattack
T


powerlaw_cluster--classic-wang_rong alph=0.5 red:HL blue:LL green:ML rep20
CFattack
T


powerlaw_cluster--Closeness Current Flow alph=0.5 red:HL blue:LL green:ML rep20
CFattack
T

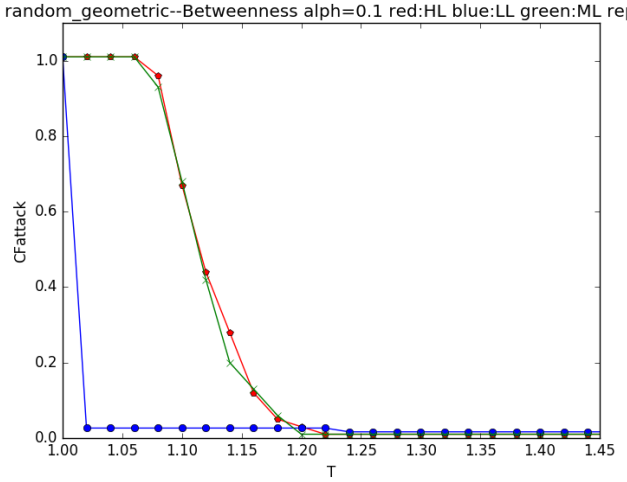

powerlaw_cluster--communicability alph=0.5 red:HL blue:LL green:ML rep20
CFattack
T


Powerlaw_cluster--Eigenvector alph=0.5 red:HL blue:LL green:ML rep20
CFattack
T

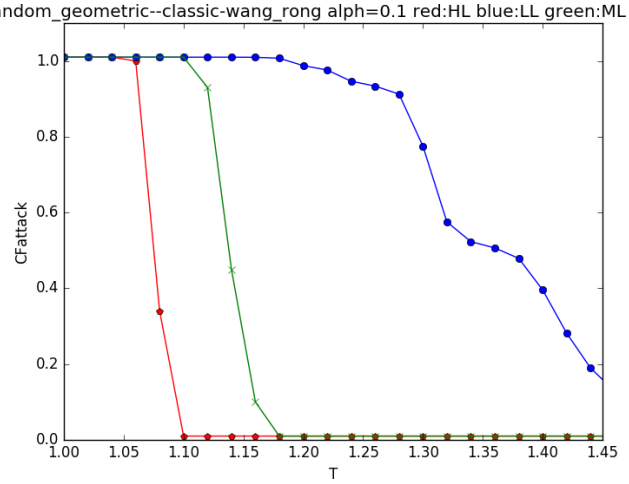

Powerlaw_cluster--Pagerank alph=0.5 red:HL blue:LL green:ML rep20
CFattack
T

Geometric 0.1

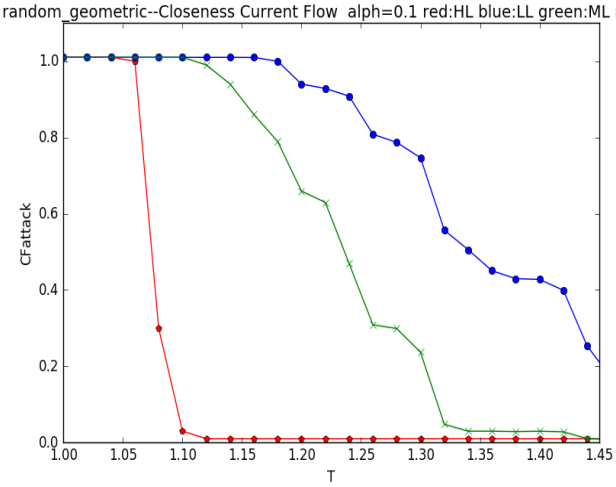
random_geometric--Betweenness alph=0.1 red:HL blue:LL green:ML rep20



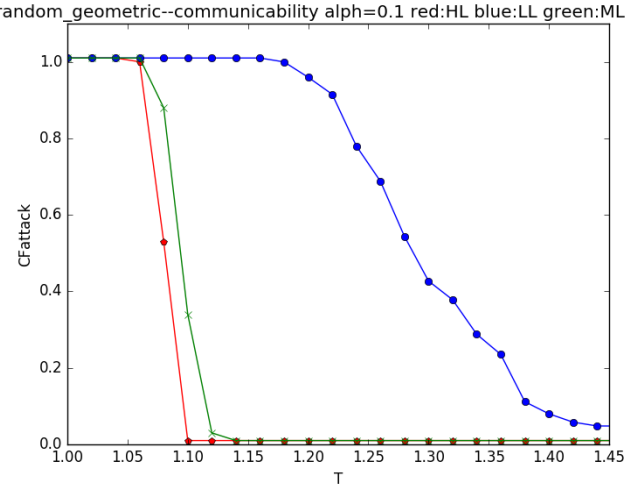
random_geometric--classic-wang_rong alph=0.1 red:HL blue:LL green:ML rep2



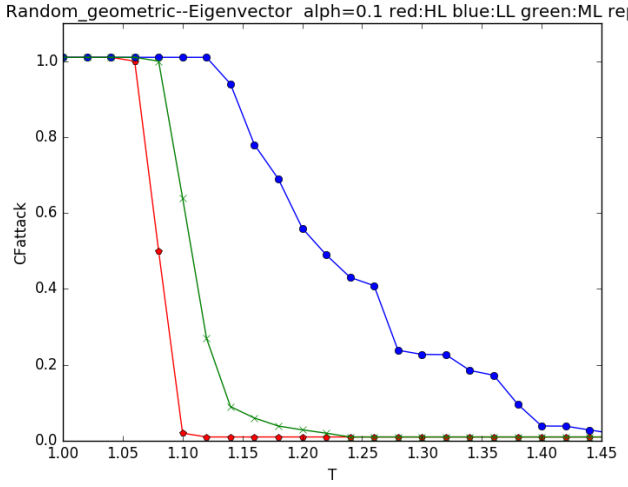
random_geometric--Closeness Current Flow alph=0.1 red:HL blue:LL green:ML rep20



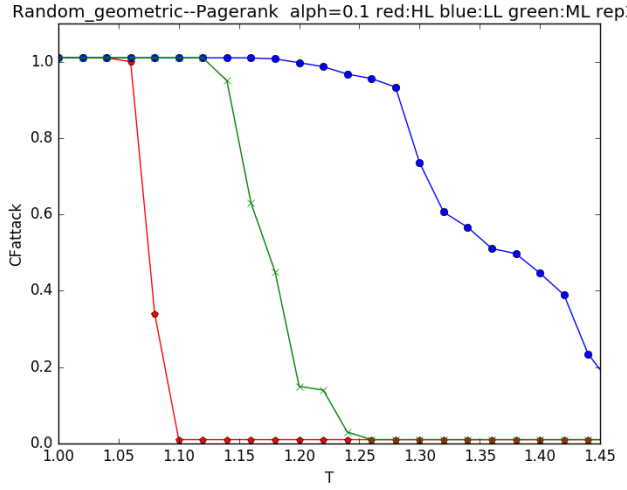
random_geometric--communicability alph=0.1 red:HL blue:LL green:ML rep20



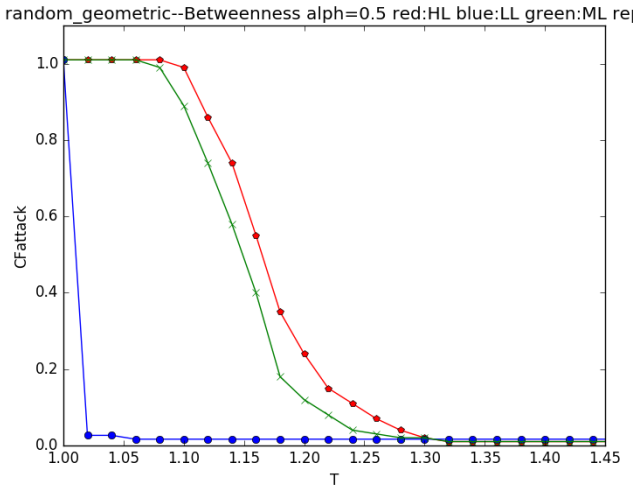
Random_geometric--Eigenvector alph=0.1 red:HL blue:LL green:ML rep20

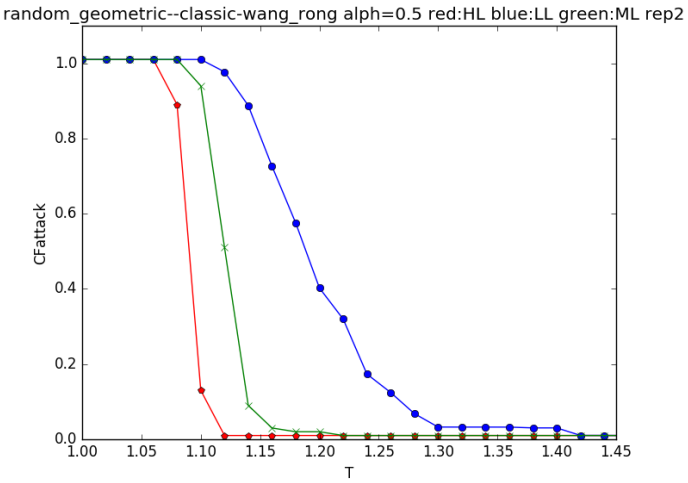


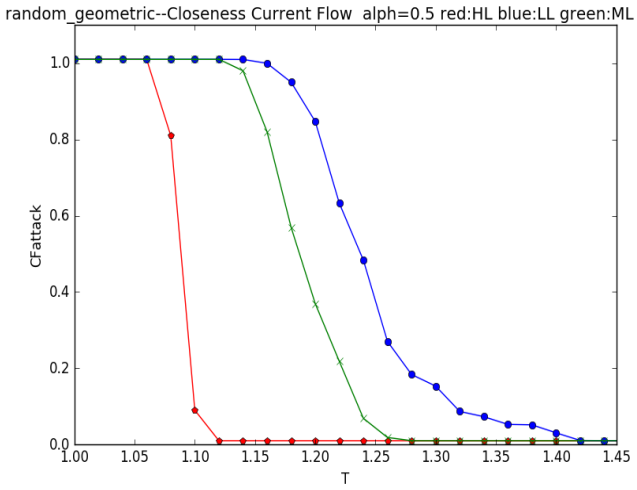
Random_geometric--Pagerank alph=0.1 red:HL blue:LL green:ML rep20

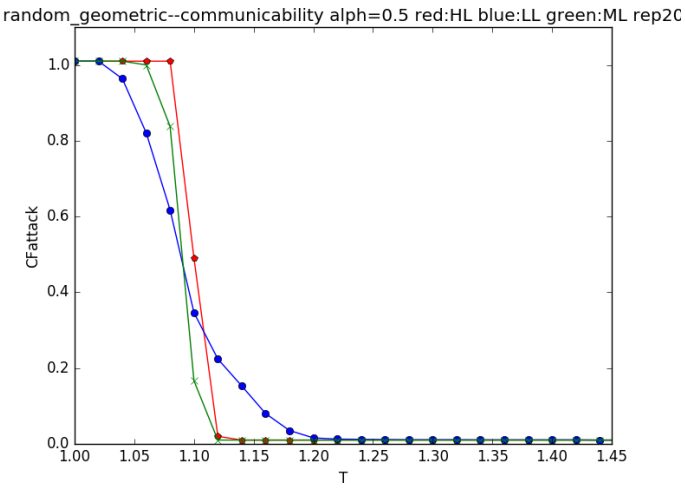


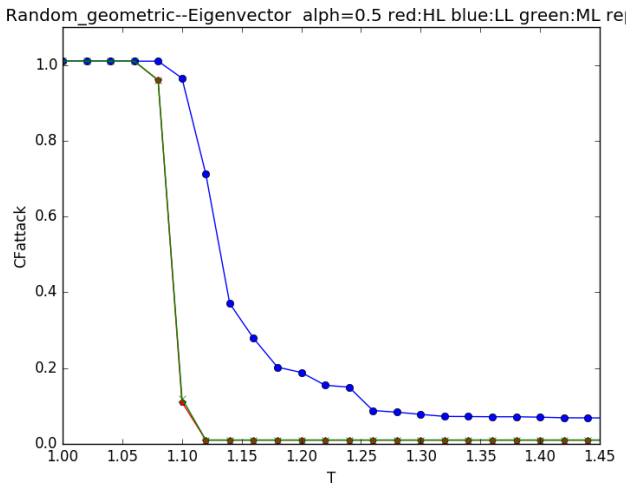
Geometric 0.5

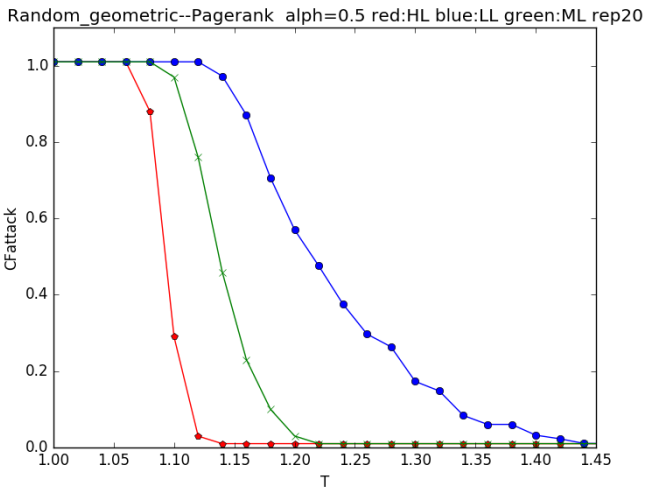




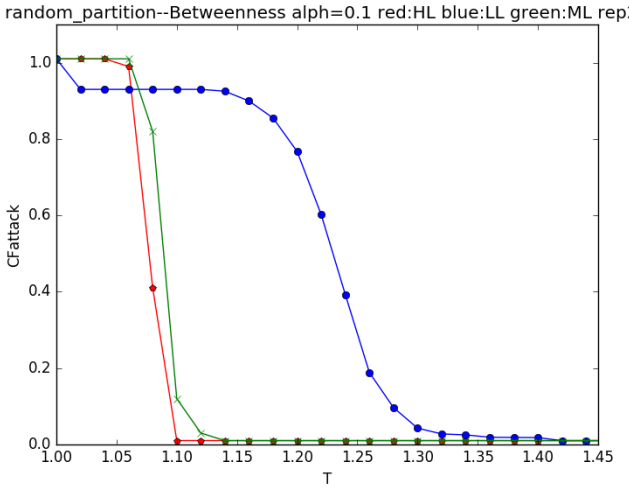


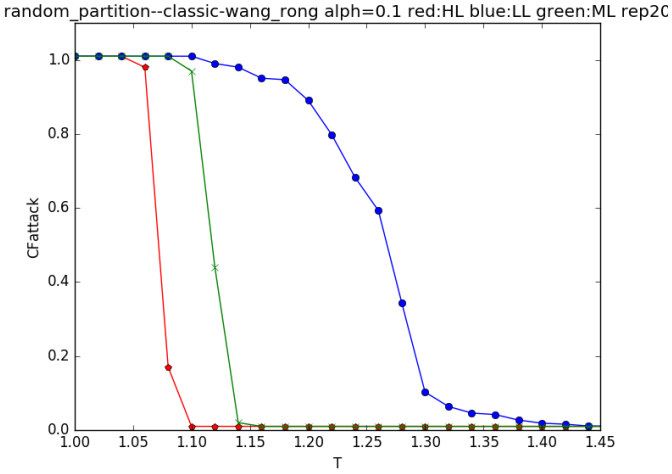


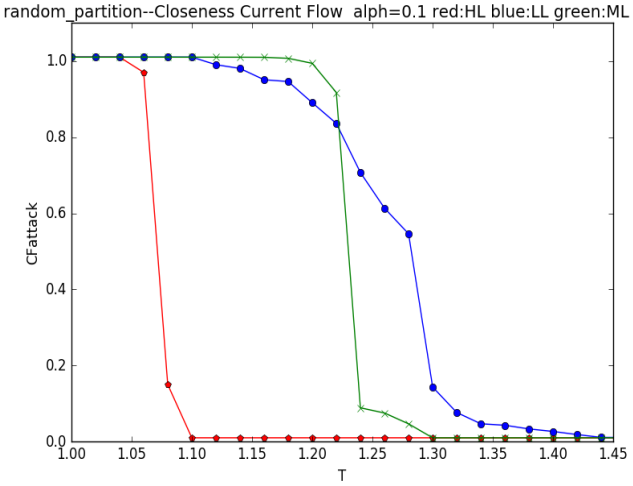


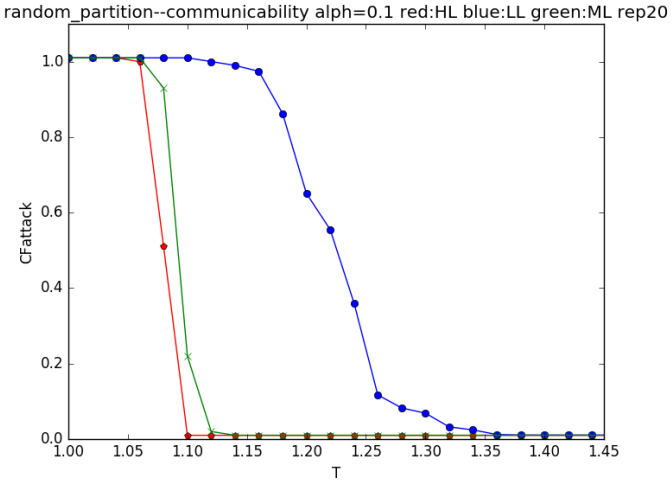


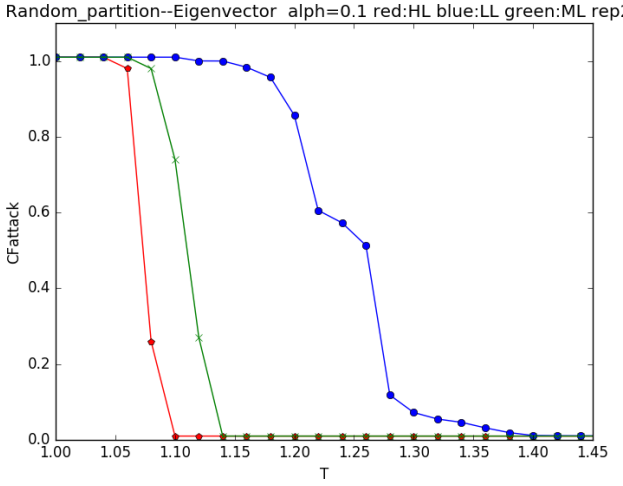
Partition 0.1

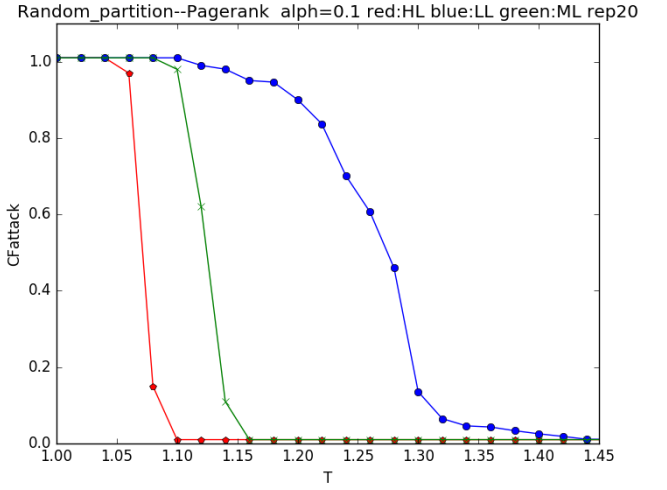




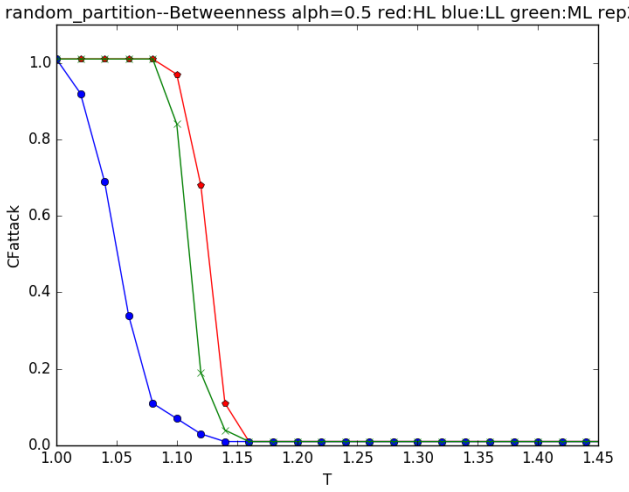


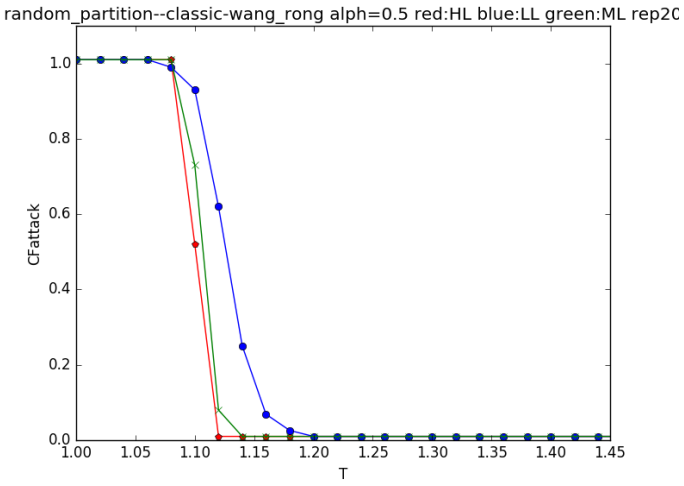


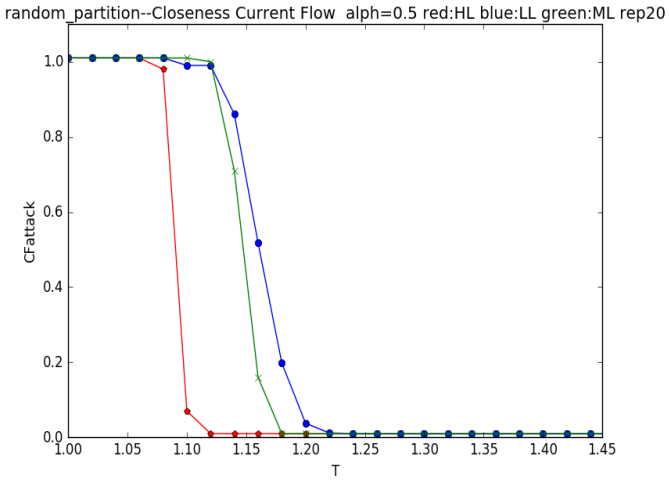


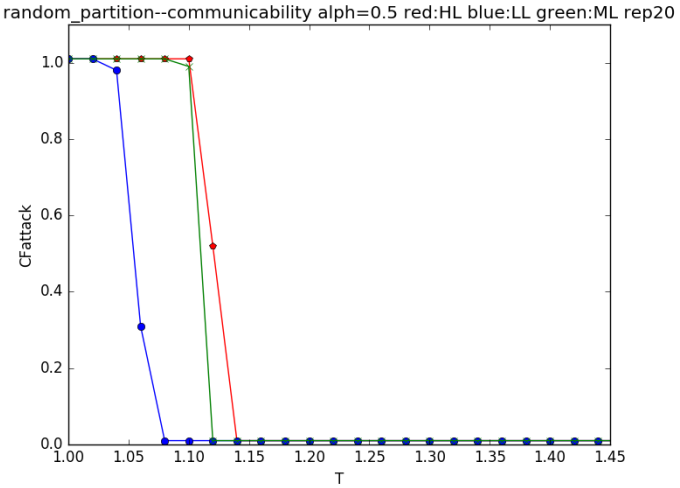


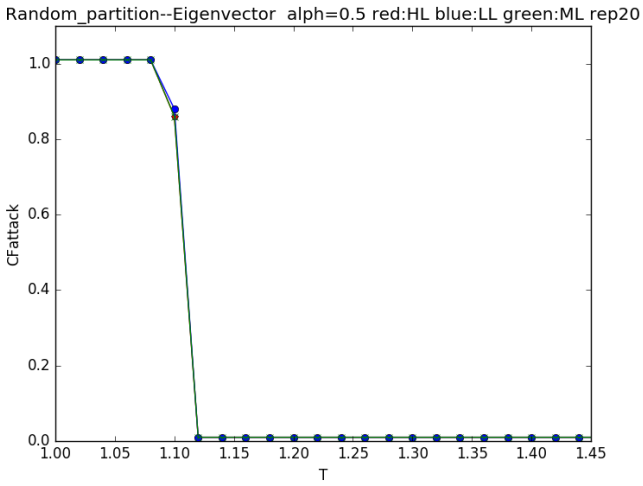
Partition 0.5

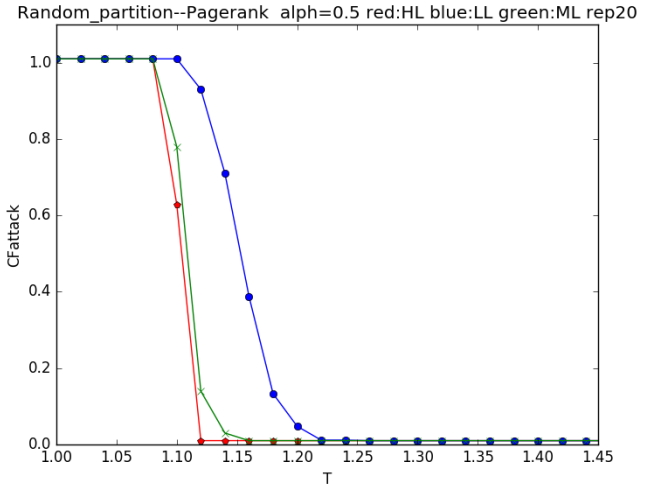

random_partition--Betweenness alph=0.5 red:HL blue:LL green:ML rep20
CFattack
T


random_partition--classic-wang_rong alph=0.5 red:HL blue:LL green:ML rep20
CFattack
T


random_partition--Closeness Current Flow alph=0.5 red:HL blue:LL green:ML rep20
CFattack
T

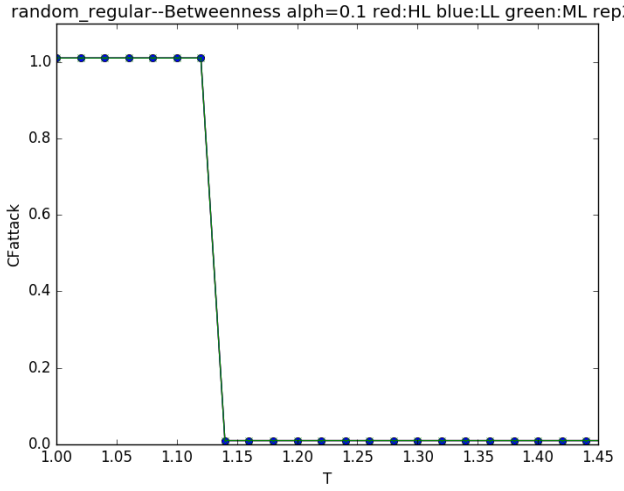

random_partition--communicability alph=0.5 red:HL blue:LL green:ML rep20
CFattack
T


Random_partition--Eigenvector alph=0.5 red:HL blue:LL green:ML rep20
CFattack
T

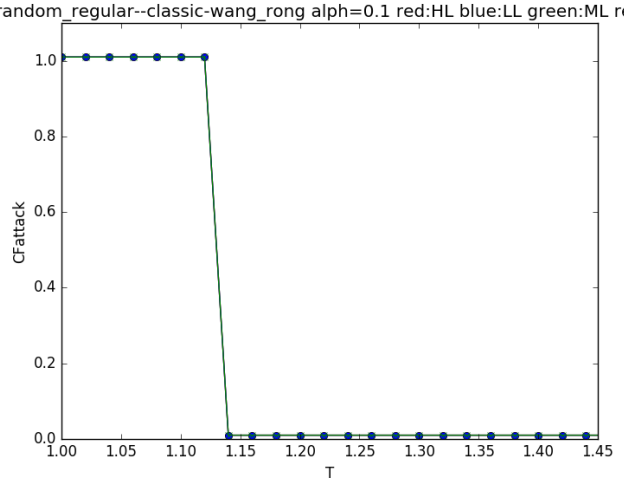

Random_partition--Pagerank alph=0.5 red:HL blue:LL green:ML rep20
CFattack
T

Regular 0.1

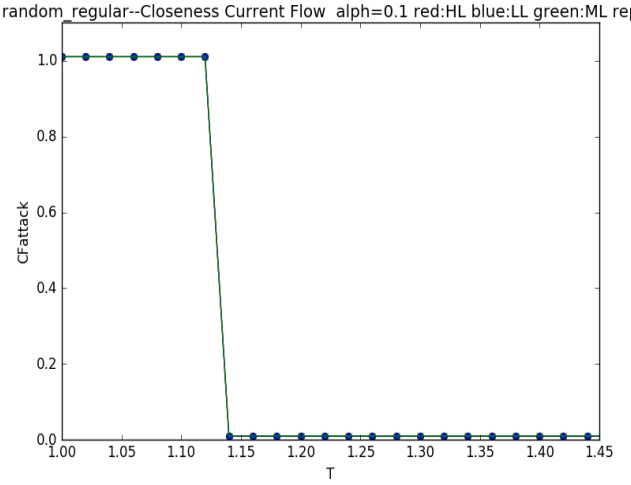
random_regular--Betweenness alph=0.1 red:HL blue:LL green:ML rep20



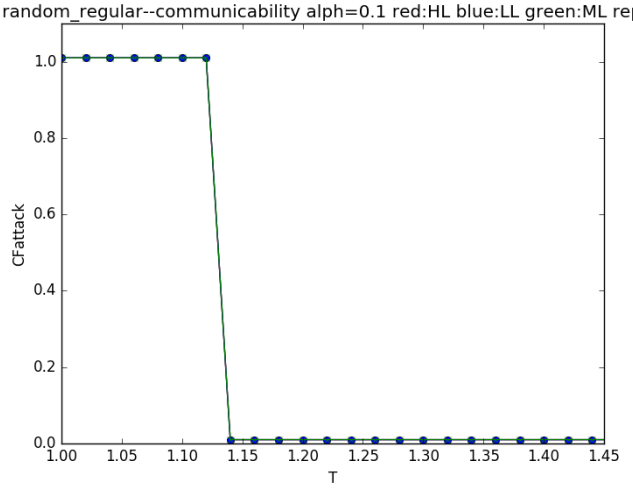
random_regular--classic-wang_rong alph=0.1 red:HL blue:LL green:ML rep20



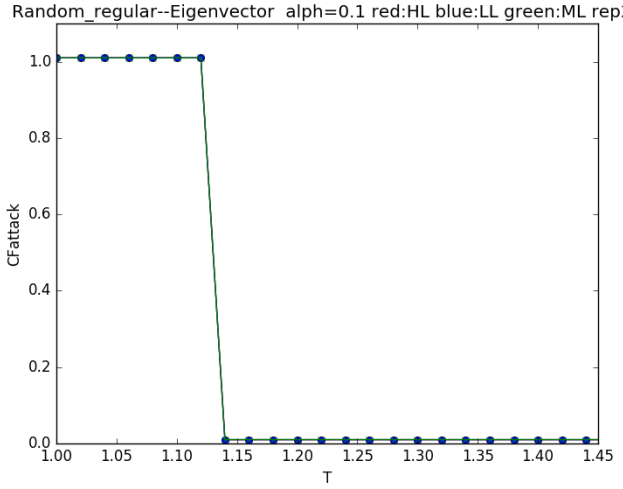
random_regular--Closeness Current Flow alph=0.1 red:HL blue:LL green:ML rep20



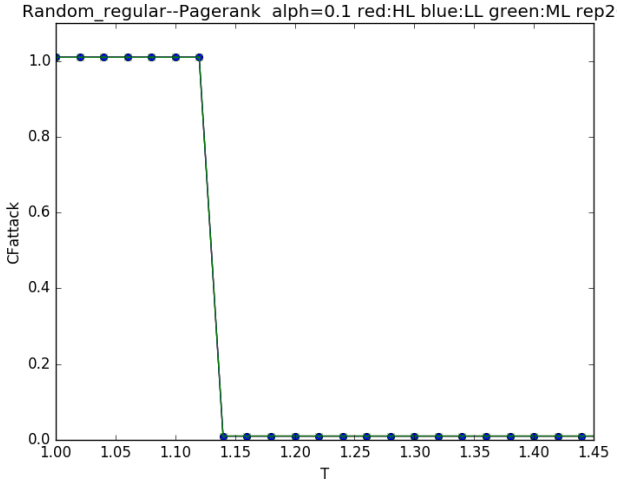
random_regular--communicability alph=0.1 red:HL blue:LL green:ML rep20



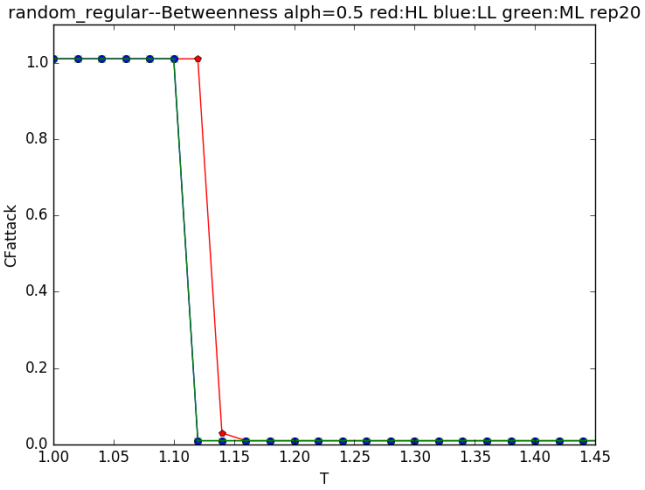
Random_regular--Eigenvector alph=0.1 red:HL blue:LL green:ML rep20

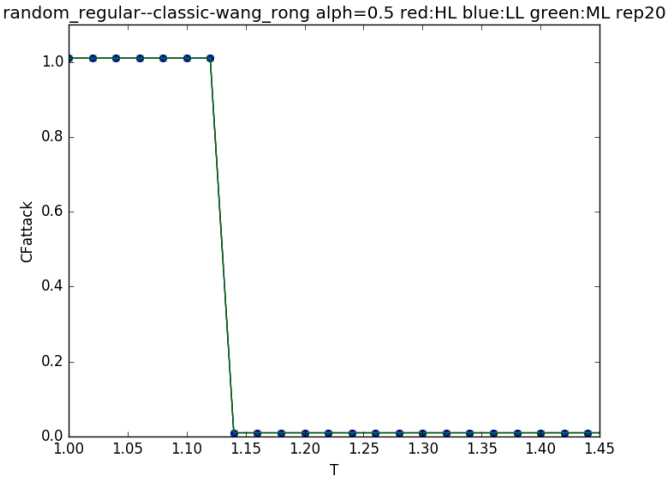


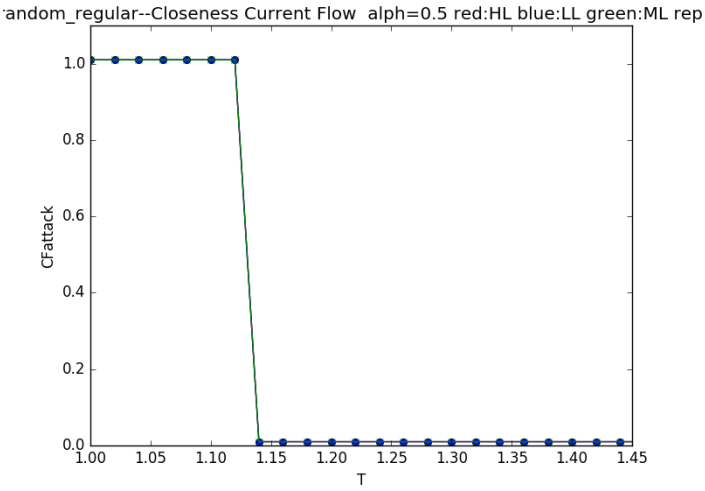
Random_regular--Pagerank alph=0.1 red:HL blue:LL green:ML rep20

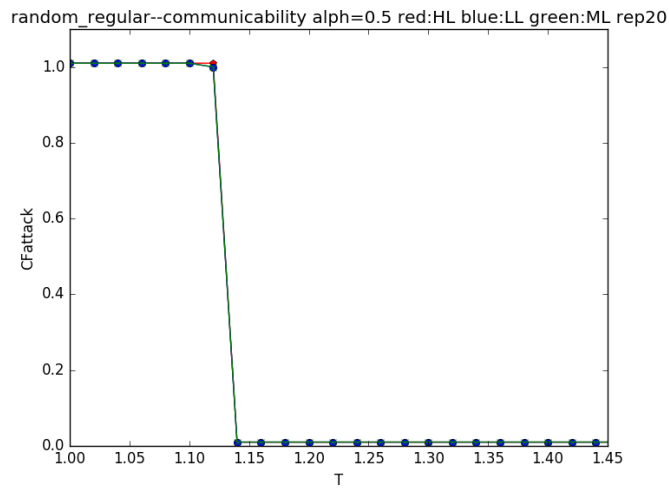


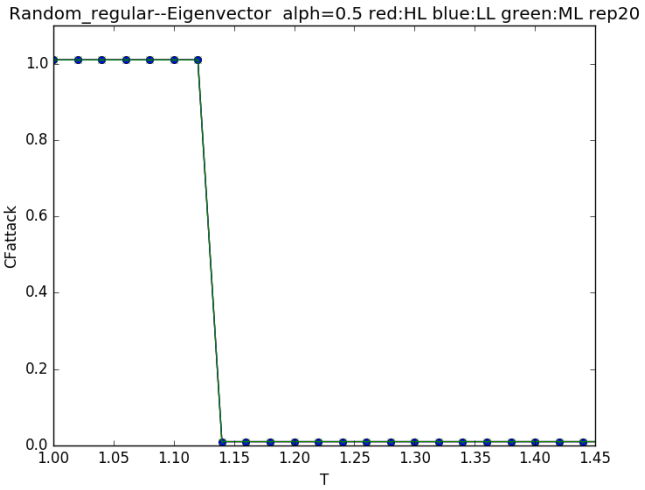
Random 0.5

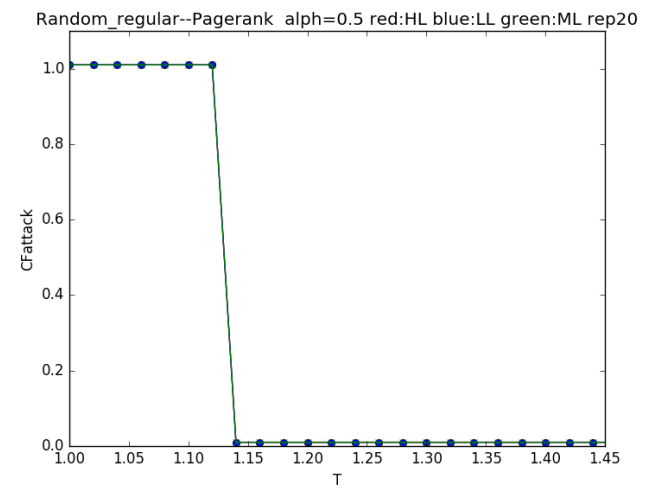

random_regular--Betweenness alph=0.5 red:HL blue:LL green:ML rep20
CFattack
T


random_regular--classic-wang_rong alph=0.5 red:HL blue:LL green:ML rep20
CFattack
T


andom_regular--Closeness Current Flow alph=0.5 red:HL blue:LL green:ML rep
CFattack
T

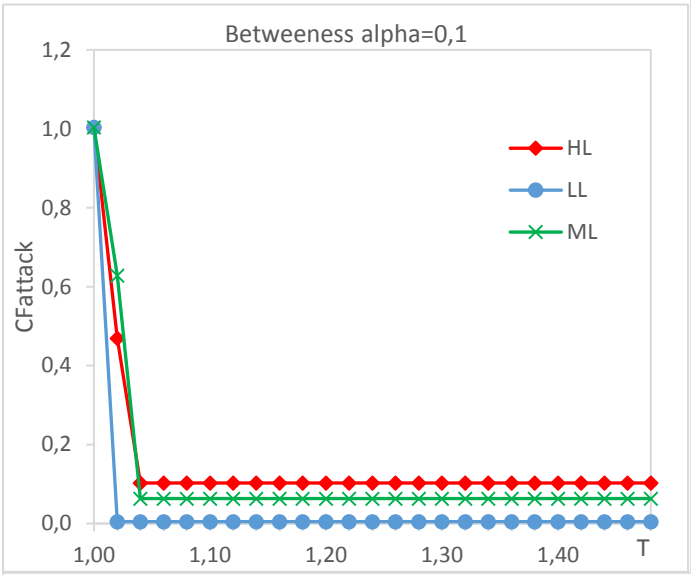

random_regular--communicability alph=0.5 red:HL blue:LL green:ML rep20
CFattack
T

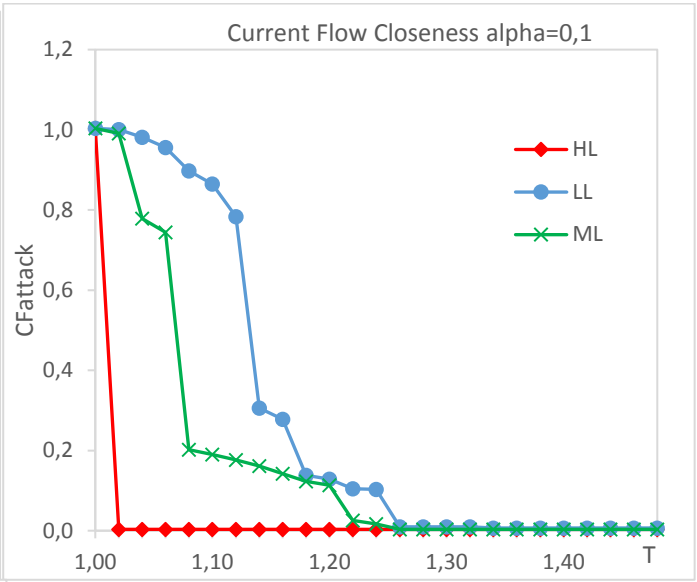

Random_regular--Eigenvector alph=0.5 red:HL blue:LL green:ML rep20
CFattack
T

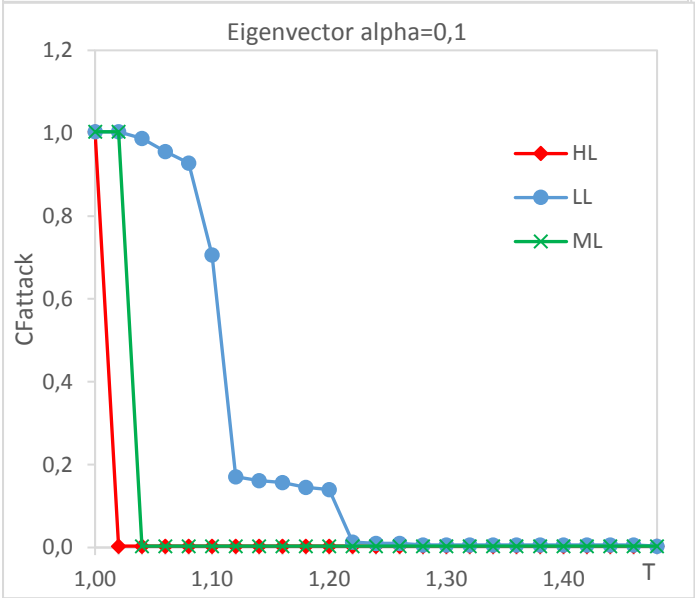

Random_regular--Pagerank alph=0.5 red:HL blue:LL green:ML rep20
CFattack
T

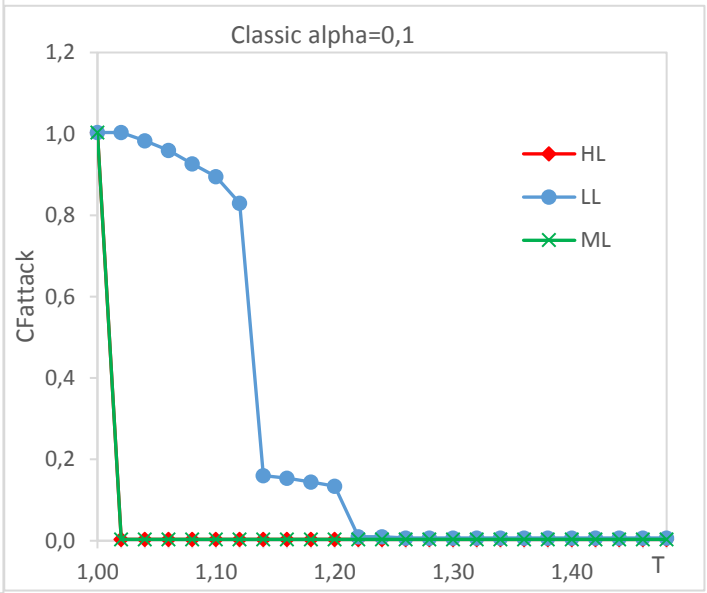
6.4. Annex D – USA plots

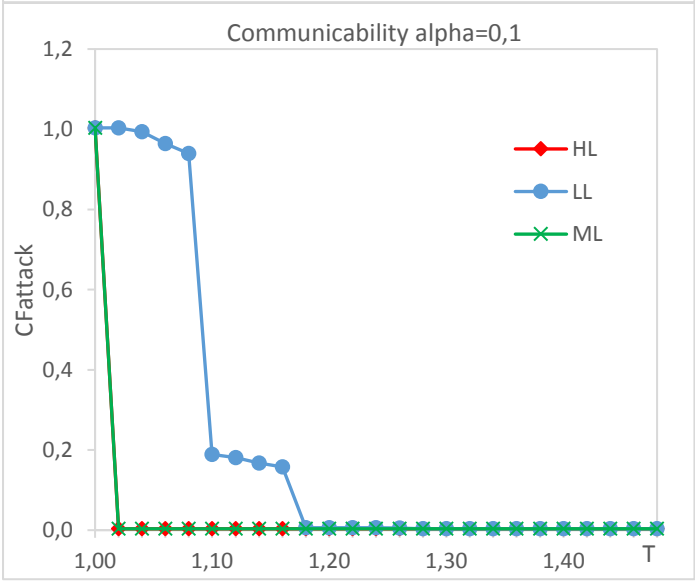
6.4.1 Sorted by Centrality

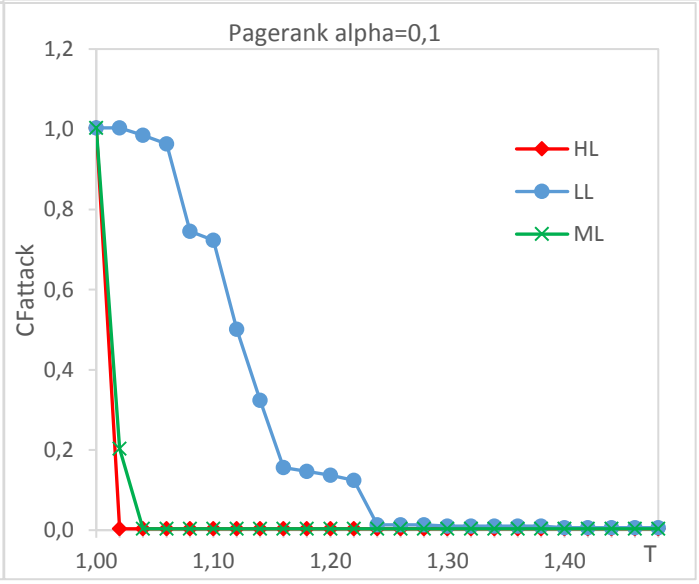

Betweeness alpha=0,1
CFattack
T
HL
LL
ML
1,2
1,0
0,8
0,6
0,4
0,2
0,0
1,00
1,10
1,20
1,30
1,40

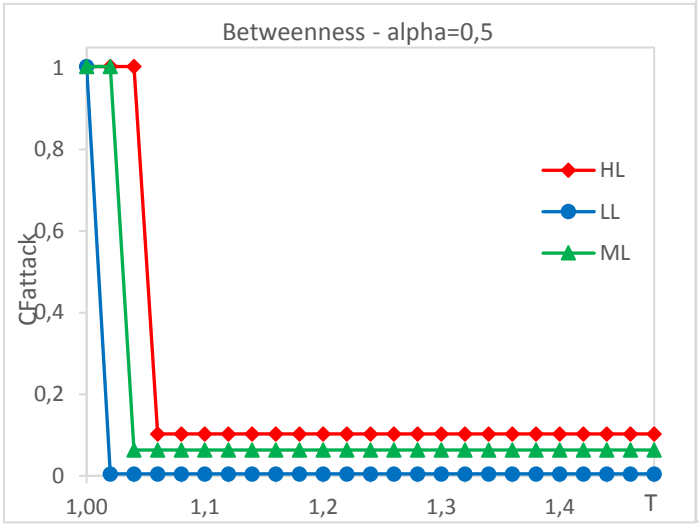

Current Flow Closeness alpha=0,1
CFattack
T
HL
LL
ML
1,2
1,0
0,8
0,6
0,4
0,2
0,0
1,00
1,10
1,20
1,30
1,40

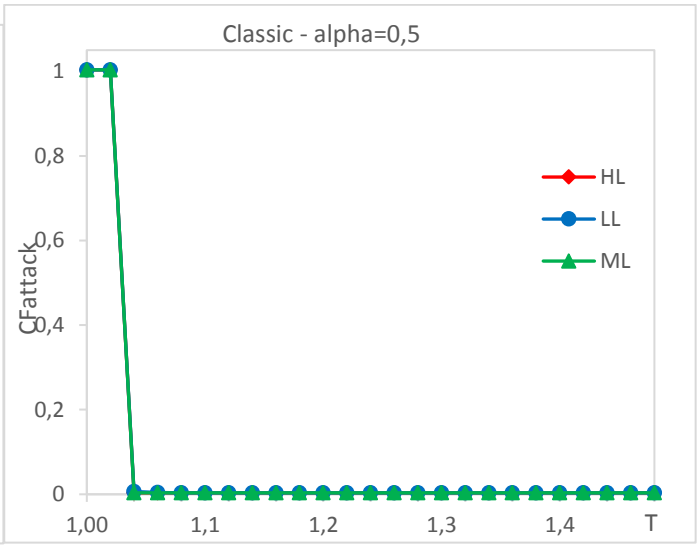

Eigenvector alpha=0,1
CFattack
T
HL
LL
ML
1,2
1,0
0,8
0,6
0,4
0,2
0,0
1,00
1,10
1,20
1,30
1,40

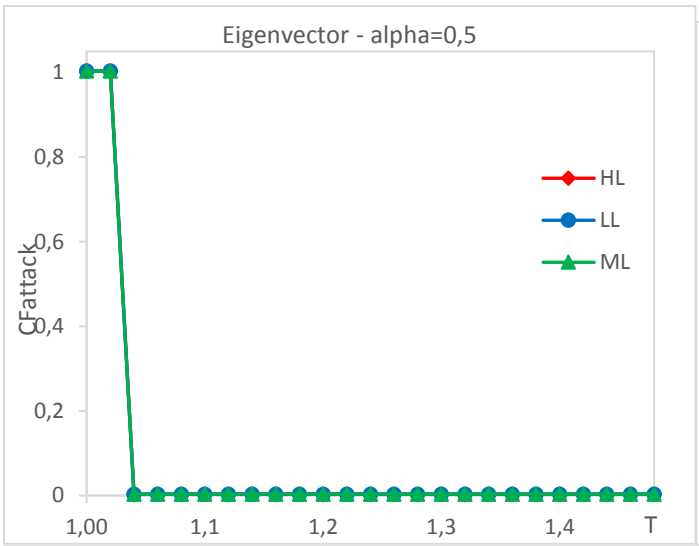

Classic alpha=0,1
CFattack
T
HL
LL
ML
1,2
1,0
0,8
0,6
0,4
0,2
0,0
1,00
1,10
1,20
1,30
1,40

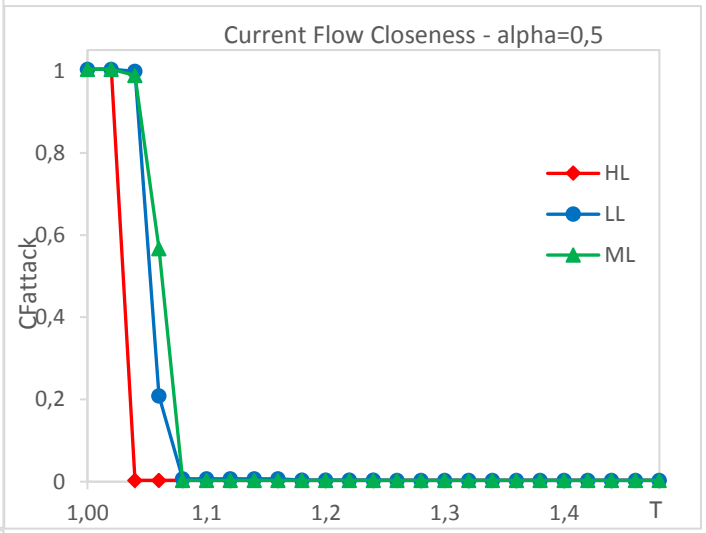

Communicability alpha=0,1
CFattack
T
HL
LL
ML
1,2
1,0
0,8
0,6
0,4
0,2
0,0
1,00
1,10
1,20
1,30
1,40

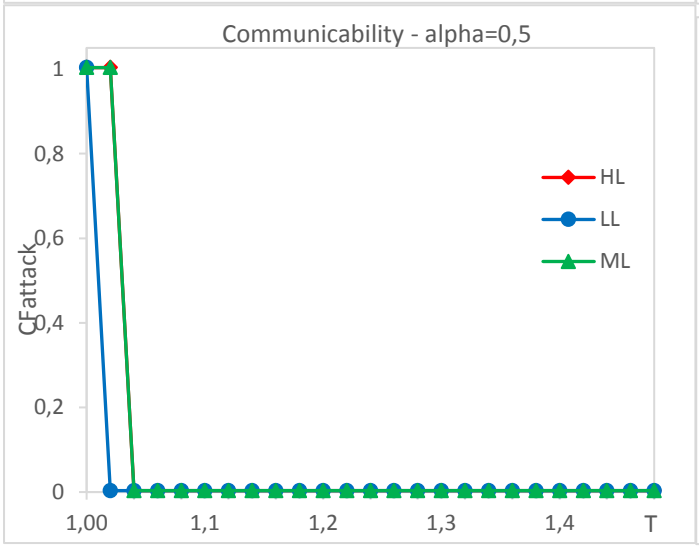

Pagerank alpha=0,1
CFattack
T
HL
LL
ML
1,2
1,0
0,8
0,6
0,4
0,2
0,0
1,00
1,10
1,20
1,30
1,40

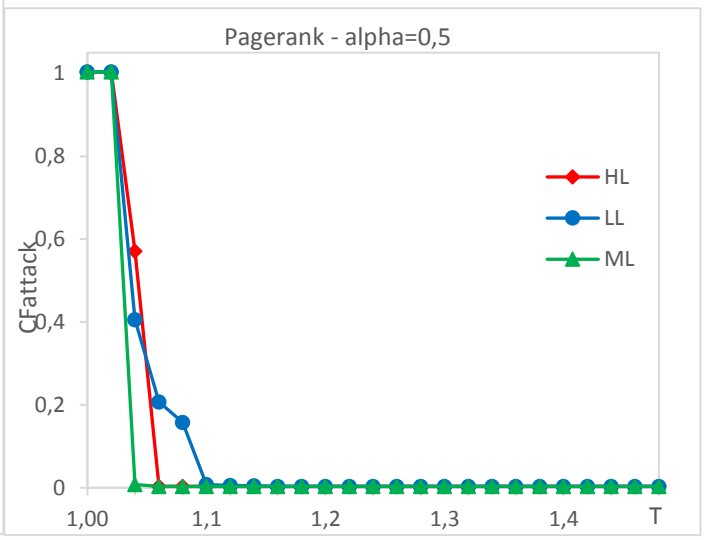

Betweenness - alpha=0,5
CFattack
T
HL
LL
ML
1
0,8
0,6
0,4
0,2
0
1,00
1,1
1,2
1,3
1,4


Classic - alpha=0,5
CFattack
T
HL
LL
ML
1
0,8
0,6
0,4
0,2
0
1,00
1,1
1,2
1,3
1,4

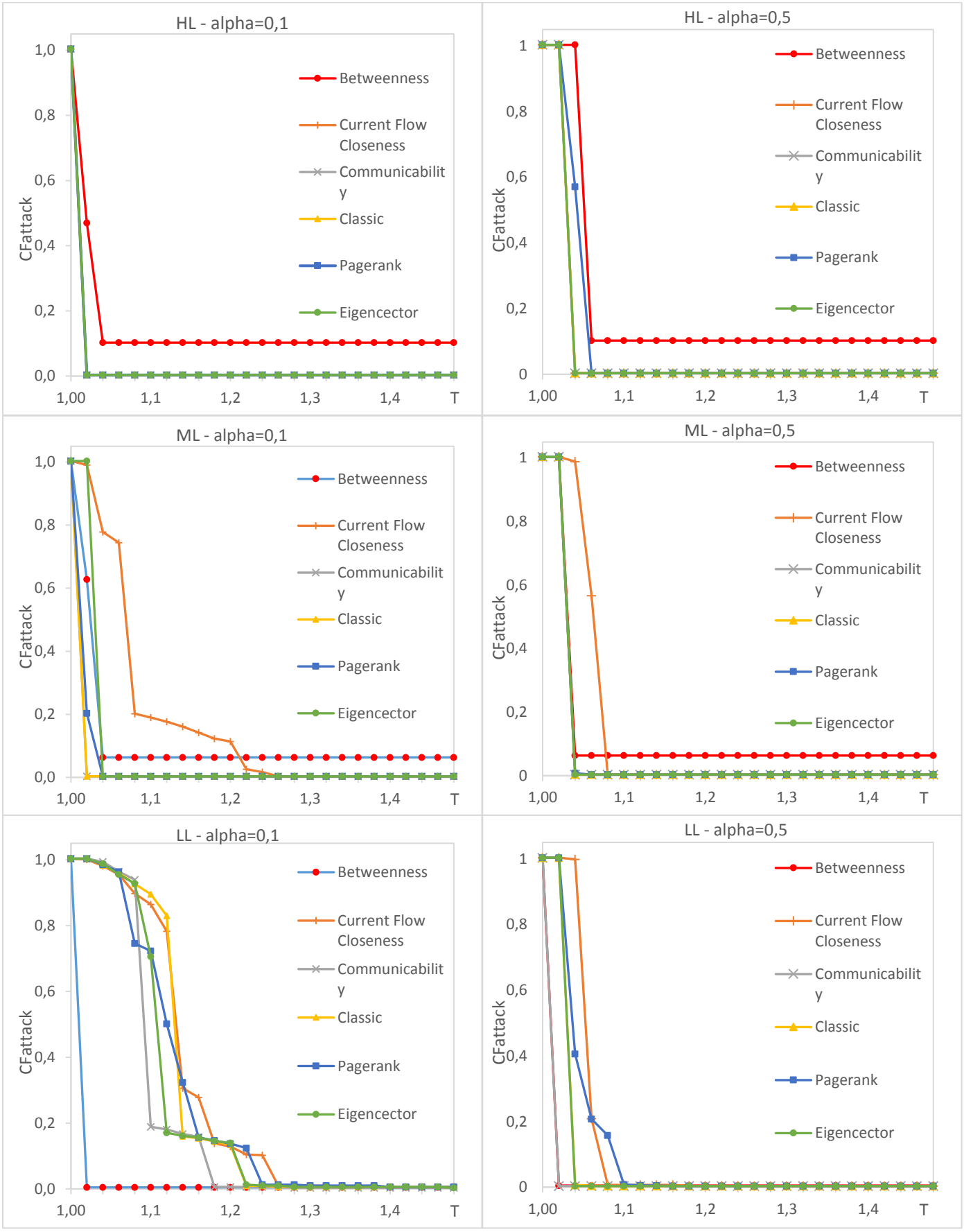

Eigenvector - alpha=0,5
CFattack
T
HL
LL
ML
1
0,8
0,6
0,4
0,2
0
1,00
1,1
1,2
1,3
1,4


Current Flow Closeness - alpha=0,5
CFattack
T
HL
LL
ML
1
0,8
0,6
0,4
0,2
0
1,00
1,1
1,2
1,3
1,4


Communicability - alpha=0,5
CFattack
T
HL
LL
ML
1
0,8
0,6
0,4
0,2
0
1,00
1,1
1,2
1,3
1,4


Pagerank - alpha=0,5
CFattack
T
HL
LL
ML
1
0,8
0,6
0,4
0,2
0
1,00
1,1
1,2
1,3
1,4

6.4.2 Sorted by Node Load Group



7 Bibliography

[1] Cohen, R., Havlin, S. (2010). *Complex Networks: Structure, Robustness and Function*. New York: Cambridge University Press.

[2] Estrada, E. (2011). *The Structure of Complex Networks: Theory and Applications*. New York: Oxford University Press Inc.

[3] Comellas, F. (2007). Models deterministes de xarxes complexes. *Butlletí de la Societat Catalana de Matemàtiques, 22* (1), 23-43.

[4] Wang, J.-W., Rong, L.-L. (2009). *Cascade Based Attack Vulnerability on the US Grid*. Safety Science, 47 (10), 1332-1336.

[5] Newma, M. (2010). *Networks: An Introduction*. New York: Oxford University Press Inc.

[6] Masuda, N., Miwa, H., Konno, N. (2005). *Geographical threshold graphs with small-world and scale-free properties*. Physical Review E 71, 036108 http://www.naokimasuda.net/papers/MasudaMiwaKonno2005PhysRevE.pdf

[7] Bradonjić, M., Hagberg, A., Percus, A. (2007). Giant component and connectivity in geographical threshold graphs, in Algorithms and Models for the Web-Graph (WAW 2007), Antony Bonato and Fan Chung (Eds), pp. 209–216, 2007. http://math.lanl.gov/~hagberg/Papers/giant.pdf

[8] <<Barabási–Albert model - Wikipedia, the free encyclopedia>> https://en.wikipedia.org/wiki/Barab%C3%A1si%E2%80%93Albert_model

[9] <<Erdös-Rényi model - Wikipedia, the free encyclopedia>> https://en.wikipedia.org/wiki/Erd%C5%91s%E2%80%93R%C3%A9nyi_model

[10] Holme,P., Kim, B.J. “Growing scale-free networks with tunable clustering”, *Physical Review E*. 65 (026107), 1-4 (2002). http://www.uvm.edu/pdodds/files/papers/others/2002/holme2002a.pdf

[11] <<Random Geometric graph - Wikipedia, the free encyclopedia>> https://en.wikipedia.org/wiki/Random_geometric_graph

[12] Dall, J., Christensen, M. “Random Geometric Graphs”, Physics Review E. 66 (016121), 1-16 (2008) https://arxiv.org/pdf/cond-mat/0203026.pdf

[13] Fortunato, S., “Complex Networks and Systems Lagrange Laboratory: Community detection in graphs”, ISI Foundation (2010) https://arxiv.org/pdf/0906.0612.pdf

[14] Estrada, E., Rodríguez-Velázquez, J.A., “Subgraph Centrality in complex networks” Phys. Rev. E, 71 (056103), (2005) https://arxiv.org/ftp/cond-mat/papers/0504/0504730.pdf

[15] Estrada, E., Fox, M., Higham, D.J., Oppo, G. *Network Science: Complexity in Nature and Technology*. Springer Science & Business Media, London, 2010.

[16] Wormald, N.C., “Models of random regular graphs”, Surveys in Combinatorics, 267, 239-298 (1999) https://www.math.uwaterloo.ca/~nwormald/papers/regsurvey.pdf

[17] Rahman, M., “Facts about random regular graphs”, MIT slides. (2016) http://math.mit.edu/~mustazee/courses/S16/beamertemp.pdf

[18] <<Random Regular graph - Wikipedia, the free encyclopedia>> https://en.wikipedia.org/wiki/Random_regular_graph

[19] Newman, M. E. J. "Modularity and community structure in networks". Proceedings of the National Academy of Sciences of the United States of America. 103 (23): 8577–8696. (2006). https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1482622/

[20] Bullmore, E. T., Sporns O. “Complex brain networks: Graph theoretical analysis of structural and functional systems” Nature Reviews Neuroscience (March 2009). https://www.researchgate.net/profile/Olaf_Sporns/publication/23974889_Complex_brain_networks_Graph_theoretical_analysis_of_structural_and_functional_systems/links/56a2476b08ae1b65112c86e9/Complex-brain-networks-Graph-theoretical-analysis-of-structural-and-functional-systems.pdf

[21] <<NetworkX documentation>> https://networkx.github.io/documentation/networkx-1.11/index.html

[22] <<PageRank Centrality>> https://www.sci.unich.it/~francesc/teaching/network/pagerank

[23] Albert R., Jeong H., Barabasi A.L., *“Error and attack tolerance of complex networks”* Nature Vol 406. (July 2000), p.378 http://barabasi.com/f/77.pdf

[24] <<Eigenvector centrality>> https://en.wikipedia.org/wiki/Eigenvector_centrality