University of Modena and Reggio Emilia

Simulation of *Highly/Heuristically Optimized/Organized Tolerance/Tradeoffs (HOT)* Networks with NetLogo

Distributed Software Systems

Supervisor: Prof. Franco Zambonelli

Author: Tomasini Marcello

Accademic Year 2011/2012

Contents

List of Figures

List of Tables

Code Listings

Chapter 1

Introduction

One of the biggest scientific and technologic challenge is develop a complete and rigorous understanding of the behavior of complex and interconnected systems. Alternative approaches to modeling the network often make extremely different assumptions and derive opposite conclusions about fundamental properties of one and the same system.
One of the most successful model in the study of complex networks are *scale-free (SF)* graphs, originally introduced by Barabási and Albert [1], which have been proposed as an universal and generic model of topology having a power-law distribution in node connectivity. Ubiquity and pervasively of this kind of distribution in natural and artificial systems make SF graphs a representative model of diverse complex systems going from social science to molecular biology till Internet.
Highly/Heuristically Optimized/Organized Tolerance/Tradeoffs (HOT) [2, 3] is the other major class of models able to predict power-law distributions. In HOT models power-law distribution results from optimization of design objectives (e.g. high performance) in presence of system uncertainty and constraints which brings to systems robust towards predicted perturbation but really sensitivity to everything not predicted. This kind of *Robust Yet Fragile (RYF)* behavior is a characteristic of complex systems in biology and engineering.

A popular case study of complex networks is internet having the main problem in how design choice and evolution have made it RYF. A line of research depict Internet as a SF network with a central hub-like structure which makes it both robust to random node losses and fragile to attacks to hub nodes. However, a more realistic model is HOT because it origins the same power-law distribution (so the hubs) but it can explain where the properties of complex organized and optimized systems come from.

1.1 Scope

In the following it will be explained how to implement a HOT network through the use of NetLogo environment [6].
In the project it will be developed an algorithm based on Fabrikant HOT model [2], underling Doyle results on the same kind of networks [4].
For a detailed and rigorous mathematical approach see [5].

1.2 Document Structure

The document is divided in 5 chapters, a short description is provided:

Chapter 2, Fundamentals: show theoretic aspects of the project and characteristics of developing environment with a lot of citations for in depth knowledge.

Chapter 3, HOT Networks: ideas, solutions and limits of the model.

Chapter 4, Implementation: detailed description of developed software.

Chapter 5, Test & Results: series of simulations to test correct behavior of the model and what are the main results.

Appendix A, Run Simulations: how to run simulations.

Chapter 2

Groundwork

The first part of the chapter gives the needed background to clearly understand the meaning and differences of SF and HOT networks. In particular some basic math definitions and results are proposed. Since the most common representation of networks are graphs, it will be used the relative definitions and it will be considered the minimum cultural background to understand the subsequent exposition. Second part introduce development environment.

2.1 Power Law and "Scale Free" Models

Power Law distribution or "Pareto's distribution" has been observed in the past century on income distribution [7], cities' population[8, 9], words' frequency [10] and in a lot of other domains, included WWW (World Wide Web)[11].

What does it mean that a network follow a Power Law distribution among node connectivity?
Given a connected simple graph G (i.e. no self-loops or parallel edges) which represent the network and has n vertices, d_i denotes the *degree* of vertex i, $1 \leq i \leq n$, that is the number of links to other vertices of the graph. The degree sequence $D = \{d_1, d_2, \ldots, d_n\}$ is how graph edges are spreaded on the vertices and it can be assumed without loss of generality always to be ordered: $d_1 \geq d_2 \geq \ldots \geq d_n$. Then, a graph G has a scaling degree sequence D (or D is scaling) if the probability P of a vertex to have d_k edges is:

$$P(d_k) = \alpha d_k^{-\gamma}$$

That is, if $\forall\ 1 \leq k \leq n$, D satisfies a power-law *rank-size relationship* of the form $k d_k^{\gamma} \approx \alpha$, where $\alpha > 0$ and $\gamma > 0$ are costants [12]. This imply: $\log P(d_k) = \log \alpha - \gamma \log d_k$ or equivalent $\log k = \log \alpha - \gamma \log d_k$; doubly logarithmic plots of $P(d_k)$ versus d_k or d_k versus rank k yield approximately straight lines of slope $-\gamma$

(fig. 2.1), while exponential rank-size relationships result in approximately straight lines on semi-logarithmic plots.

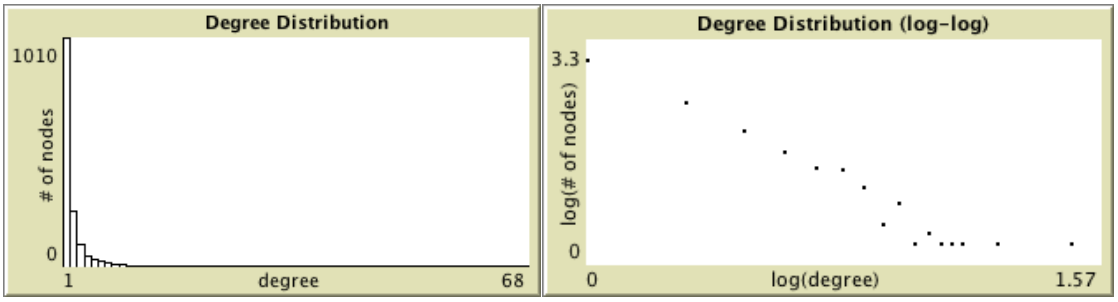

Figure 2.1: Power Law Degree Sequence generated by "Preferential Attachment". Only $d_k > 1$ are shown.

It is immediate to note that power-law distribution decay as a polynomial, so it has a "long tail", that is the area under the density function in interval $[k, +\infty[$ diverge for $k \to \infty$.
Power Law distribution imply:

- infinite variance (if $\gamma \leq 2$)
- probability of elements far from the average not negligible
- big numbers "count"

Every system having a Power Law distribution follow the $80 - 20$ *Rule*. In the specific case of networks it says that 20% of nodes have 80% of links.

There have been several attempts to explain how the Power Law emerges in complex systems using " generative" models [13]; most of them fall in "scale-free growth" or "preferential attachment"[1] category, that is the growth of individuals in a population follows a stochastic process independent of the size of the individual, so that biggest individuals attract more growth (e.g. the rich get richer, nodes with more links attract other nodes to form even more links).
SF networks exhibit the presence of nodes that:

- act as hub, i.e., as a point where most of nodes is connected
- act as connectors, i.e., nodes that contribute largely to the connection of the portions of the network
- minor nodes that act as hubs or connectors to local portions of the network

Most nodes have a limited number of links instead. This structure means that, whatever the scale at which it is observed, the network appears similar to itself (fig. 2.2), that is auto-similar. Not only that, the network maintains its properties

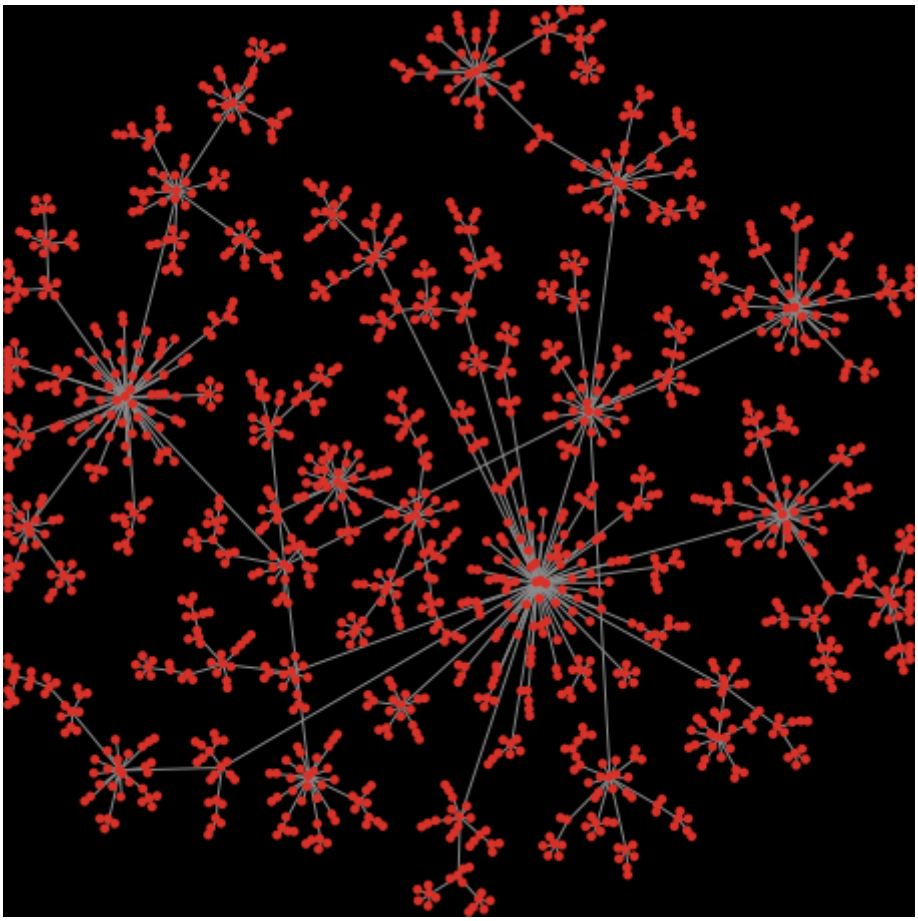

Figure 2.2: Scale Free Network generated by "Preferential Attachment".

regardless of the scale and, in particular, if you delete the details of a network, such as to skip nodes with a limited number of links, maintains its power-law structure, as well as if one considers only a portion of it will have the general structure of the whole network.

2.2 NetLogo Simulation Environment

NetLogo [6] is a programmable modeling environment for simulating natural and social phenomena based on a Logo dialect. NetLogo runs on the Java virtual machine, so it works on all major platforms (Mac, Windows, Linux, et al).

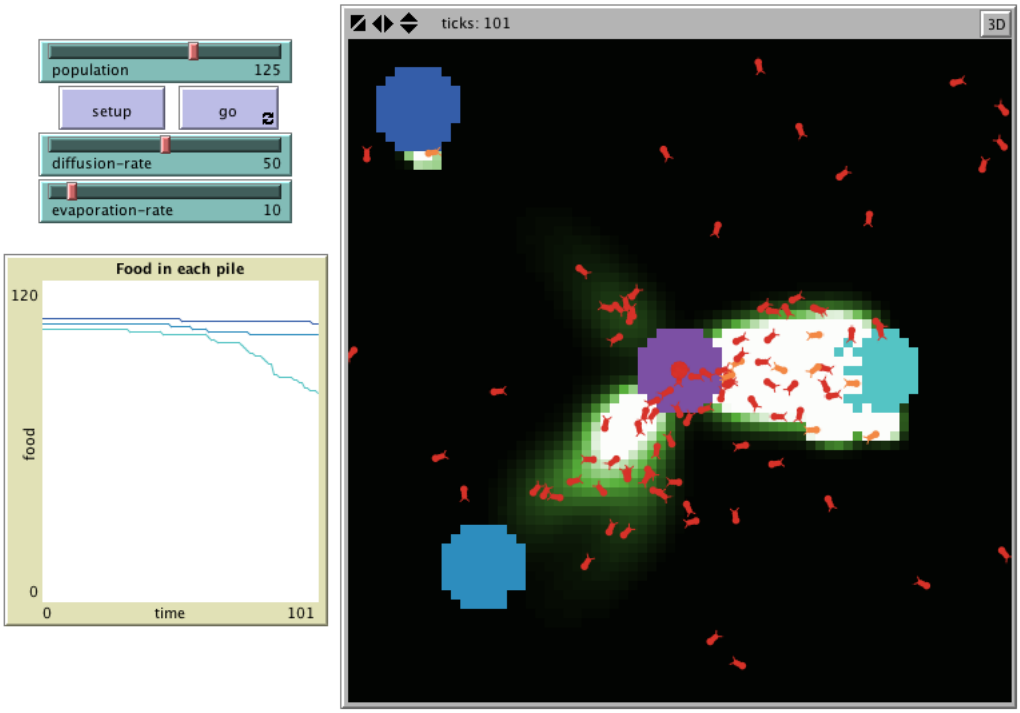

Figure 2.3: Example of natural phenomena simulation: Ants Foraging.

NetLogo is particularly well suited for modeling complex systems developing over time the nature of which is decentralized and interconnected, including network phenomena. Modelers can give instructions to hundreds or thousands of "agents" all operating independently. This makes it possible to explore the connection between the micro-level behavior of individuals and the macro-level patterns that emerge from their interaction (fig. 2.3). It also comes with the Models Library, a large collection of pre-written simulations that can be used and modified. [14, 15].

The NetLogo world is made up of agents [16]. Agents are beings that can follow instructions. In NetLogo, there are four types of agents: turtles, patches, links, and the observer. *Turtles* are agents that move around in the world. The world is two dimensional and is divided up into a grid of patches. Each patch is a square piece of "ground" over which turtles can move. *Links* are agents that connect two turtles. The observer doesn't have a location – you can imagine it as looking out over the world of turtles and patches. The *observer* doesn't observe passively – it gives instructions to the other agents. When NetLogo starts up, there are no turtles. The observer can make new turtles. Patches can make new turtles too (patches can't move, but otherwise they're just as "alive" as turtles). Patches have coordinates. The patch at coordinates (0,0) is called the origin and

the coordinates of the other patches are the horizontal and vertical distances from this one. We call the patch's coordinates `pxcor` and `pycor`. Just like in the standard mathematical coordinate plane, `pxcor` increases as you move to the right and `pycor` increases as you move up. The total number of patches is determined by the settings `min-pxcor`, `max-pxcor`, `min-pycor`, and `max-pycor`. Turtles have coordinates too: `xcor` and `ycor`. A patch's coordinates are always integers, but a turtle's coordinates can have decimals. This means that a turtle can be positioned at any point within its patch; it doesn't have to be in the center of the patch. Links do not have coordinates. Every link has two ends, and each end is a turtle. If either turtle dies, the link dies too. A link is represented visually as a line connecting the two turtles.

In NetLogo, commands and reporters tell agents what to do. A *command* is an action for an agent to carry out, resulting in some effect. A *reporter* is instructions for computing a value, which the agent then "reports" to whoever asked it. Commands and reporters built into NetLogo are called primitives. The NetLogo Dictionary has a complete list of built-in commands and reporters [17]. Commands and reporters you define yourself are called *procedures*. Each procedure has a name, preceded by the keyword `to` or `to-report`, depending on whether it is a command procedure or a reporter procedure. Once you define a procedure, you can use it elsewhere in your program.

Agent variables are places to store values in an agent. An agent variable can be a global variable, a turtle variable, a patch variable, or a link variable. If a variable is a global variable, there is only one value for the variable, and every agent can access it. You can think of global variables as belonging to the observer. Turtle, patch, and link variables are different. Each turtle has its own value for every turtle variable. Users can also define their own variables. They can make a global variable by adding a *switch*, *slider*, *chooser*, or *input box* to the model, or by using the `globals` keyword at the beginning of the code. They can also define new turtle, patch and link variables using the `turtles-own`, `patches-own` and `links-own` keywords. These variables can then be used freely in the model. Use the `set` command to set them. (Any variable don't setted has a starting value of zero). A local variable is defined and used only in the context of a particular procedure or part of a procedure. To create a local variable, use the `let` command.

In many NetLogo models, time passes in discrete steps, called *ticks*. NetLogo includes a built-in tick counter to keep track of how many ticks have passed. To retrieve the current value of the tick counter, use the `ticks` reporter. The `tick` command advances the tick counter by 1 and it will usually also update the view.

Chapter 3

HOT Networks

A mechanism to generate power-law distributions, which is inspired by how biological organisms are organized and how systems with a high degree of engineering are structured is *Highly/Heuristically Optimized/Organized Tolerance/Tradeoffs (HOT)*. The focus falls on systems that are optimized through natural selection or construction, to provide a solid performance despite a stochastic environment. The power-law distribution in these systems arises because of tradeoffs between yield, cost of resources and risk tolerance. These tradeoffs lead to highly optimized design, which evolve in a way that rewards strategies subject to specific forms of external stimuli, but occasionally allow "large" events (i.e., with a low probability to happen, but with relevant effects). This is in contrast with *SOC (Self Organized Criticality)* and *EOC (Edge Of Chaos)*, where external forces are used only to start the events, but the mechanism that causes the complexity is essentially independent; while in SF networks it is called "emerging complexity", HOT networks have "organized complexity".

3.1 Characteristics of HOT Systems

HOT is an attempt to use simple models to capture the essence of the role of design and/or evolution in the creation of highly structured configurations exhibiting power laws, self-dissimilarity, scale-richness, etc.
Highly/Heuristically Optimized/Organized alludes to the fact that the objective is reached with a configuration highly structured, non-generic and "rare", which provides high performance, even if it is not the mathematically optimal solution, but only a good approximation.
Tolerance/Tradeoffs emphasizes that the robustness in complex systems, that is the ability to maintain some of the features you want in spite of the uncertainties in the behavior of the components and of the environment, is a quantity limited and

constrained, which must be properly managed. Many of the important properties to which a system aspires can be viewed as a specific type of robustness. *Reliability* includes robustness to component failure. *Efficiency* is strength to the shortage of resources. *Scalability* is robustness to change in size and complexity of the system as a whole. *Modularity* is strength to components reorganization. *Evolvability* is robustness of lineages to changes over long time scales.

The connection between advanced technologies and biology is not accidental, since the complexity of engineering systems is approaching that of biological systems. Most of engineering and biological systems are not designed in accordance with a global optimization, but evolve through the exploration of local variations with occasional structural changes. The biological evolution makes use of a *genotype*, which can be distinguished, at least theoretically, from the *phenotype*. In engineering this distinction is cleaner, because the design specifications exist completely independently from the physical instance. In both cases, genotype can evolve due to some form of "natural selection" on the yield.
To understand the highly organized systems it is useful to note that the structure is a consequence of specific constraints on their functionality and/or on their behavior and this is largely independent from the process by which the organization grows. The organization constraints are:

- component level, in terms of what they can do and their uncertainty (e.g. level of reliability)
- system level, there are constraints on the system as a whole that are not a consequence of those on the components, including functional requirements (i.e., what the system should do), the environment and the operational requirements (e.g. conditions in which the system must operate) and robustness to certain perturbations from outside or inside the system
- protocol level, typically in the form of rules for the configuration and/or the interaction of the components in the system
- "emerging" constraints, resulting from interactions between non-trivial constraints at system level and components level. The emergence is also associated with unintended consequences, both positive and negative (e.g. fragility to specific perturbations)

The organization can be viewed as a specialized structure that allows the system to satisfy all constraints. The structure of HOT complex systems is: highly modular, often consisting of cheap and imperfect components, it has late binding to functionalities that allows evolution of diverse skills and behaviors, it can quickly and adaptively change through the use of distributed control and feedback.

Most of engineered systems starts with certain design choices, each one having its compromises, which led HOT systems have certain characteristics:

- high efficiency, performance and resilience to perturbations considered in the design
- hypersensitivity to design flaws and unexpected perturbations
- structured and specialized configuration
- power-law distribution

These features are a result of optimization of the design goals under uncertainty and constraints. The self-similar structures (e.g. SF networks) rarely fulfill the objectives of specialized design, with the exception of distribution networks, which are inherently tree-like and often fractals, while the subsystems's hierarchies in complex systems of engineering and biology have a self-dissimilar structure.

3.2 Internet: a Model of HOT Network

Presents a network "toy" model, reflecting the HOT approach in shaping the router level of Internet, in contrast with the corresponding SF, because it does not require assumptions, implicit or explicit, derived directly from sets and random processes . The compromises on the Internet can be explained without insisting on a underlying statistical model; sources of randomness are naturally included where uncertainty needs to be managed and taken into account (e.g. location of the users). It is important to note that this model does not claim to be realistic, but exemplifies the principles by which a net having a robust and performant design, which depend on objectives and constraints, can be generated.

3.2.1 Is Internet HOT?

An important feature of the highly organized complexity, although largely hidden, of Internet is to make the system as a whole robust to perturbations for which is designed [18], but also potentially vulnerable to other perturbations [19]. All components must be bound by protocols, but because of an extensive feedback control, the system can tolerate a huge variability, while respecting the constraints, and still provide a robust and reliable functionality to the applications, which are the components at most high level and therefore less constrained. Since the lack of a component is allowed, the system must be robust by design to faults or attacks that cause the removal of it. However, this robustness and adaptivity coexists with an equally high fragility to attacks targeting mechanisms that provide services to

the higher levels of the stack [20]. The understanding of the robustness of Internet requires a perspective that incorporates protocols, layering and adjustment of feedback, this view suggests that the most essential RYF characteristics arises from aspects that are only indirectly connected with the connectivity of the graph. So it can be said that the RYF nature of the Internet is the result of its highly developed and organized structure, that is HOT.

3.2.2 Constraints and Functional Objectives

A HOT model of Internet router topology require two general elements: constraints and functional objectives.
First, the technological and economic constraints on components such as routers and links and their interconnections restrict what topologies are feasible or possible. Economically speaking the cost of installation and operability of links increases with the length of the connection and can be the dominant component of the total budget, especially in the backbone. On the one hand routers impose constraints on bandwidth, on the other the cost of the links places strong incentives to minimize the number and length of the connections.
Secondly, the backbones and the connectivity of the routers are subsystems in the decentralized and layered infrastructure of Internet. The consequence is that these subsystems can be truly understood only in the light of functionality that they provide to higher levels of the protocol stack and the rest of the network.

The basic idea is that economic considerations and technological factors constrain network topology of the ISP. While SF models are universal in the sense of "general", HOT models are universal in the sense that they make a topology in terms of robustness and optimization, but with goals and constraints of a specific application domain.

3.2.3 A Generative Model

It has been observed [21] that the degrees of the graph of Internet (both of the routers that of Autonomous Systems) obey to power-law. This observation has led to a review of how to generate Internet graph model [22].

In HOT models, power laws are the result of a optimized and reliable design in the presence of constraints and uncertainty. A possible model for Internet growth, as proposed by Fabrikant [2], is to optimize two objectives simultaneously: *"last mile" connection cost*, defined as the Euclidean distance between two nodes that must be connected and *transmission cost*, measured in number of routers that packets must traverse (hops), that is, nodes try to connect "centrally" in the network, to minimize transmission delays. This simple and primitive model generates a sequence of node degrees that follows a law power-law.

The model constructs a simple connected acyclic graph (i.e., a tree) starting from a node (root), which is the core of the network, while the other nodes arrive uniformly distributed in space; node i attaches itself to the node j that minimizes the weighted sum of the two objectives:

$$min_{j<i}\ \alpha d_{ij} + h_j \tag{3.1}$$

where d_{ij} is the *normalized Euclidean distance* and h_j is some measure of the "centrality" of node j with the following meaning:

(a) the average number of hops from other nodes

(b) the maximum number of hops from another node

(c) the number of hops from a fixed center of the tree

α is a parameter, that can be seen as a function of the final number of nodes n, which measures the relative importance of the two objectives. The behavior of the model depends crucially on the value of α (theorem 1 [2]):

- if $\alpha < c$, where c is a constant that depends on the shape of the region, then Euclidean distances are not important, and the resulting network is easily seen to be a star, the ultimate in degree concentration, and, depending on how you look at it, the exact opposite, or absurd extreme, of a power law

- if $\alpha \geq \sqrt{n}$, where n is the final number of points, then Euclidean distance becomes too important, and the resulting graph is a dynamic version of the Euclidean minimum spanning tree, in which high degrees do occur, but with exponentially vanishing probability

- if $c < \alpha < \sqrt{n}$ then, almost certainly, the degrees obey a power law

It should be noted that this network model exhibits a behavior of the form "the rich get richer" typical of SF models: the nodes that come first have easily a high degree and a low cost in terms of hops and thus are more eligible to be choosen by nodes that come after. However this is not the result of a primitive assumption (and therefore difficult to defend), but rather a consequence of local optimization to meet certain constraints. It is possible to extend the model to more general graphs, attacking the new nodes to some of the most beneficial nodes, where the number of new edges is appropriately distributed to produce graphs with the correct average degree > 1. This results in a particularly interesting model, since satisfies a number of properties of the Internet as well as the degrees distribution of vertices [23].

3.3 S Metric

For a graph g having degree sequence $D = \{d_1, d_2, \ldots, d_n\}$ it is possible to define the purely graph-theoretic quantity ([5], capitolo 4):

$$s(g) = \sum_{(i,j)\in E(g)} d_i d_j \tag{3.2}$$

where $E(g)$ is the set of edges in the graph. It is easy to check that high $s(g)$ requires high-degree vertices to connect to other high-degree vertices.
Normalizing against $s_{max} = max\{s(g) : g \in G(D)\}$, where $G(D)$ denotes the set of all simple connected graphs having degree sequence D, it can be defined the measure $0 \leq S(g) \leq 1$ of the graph g as:

$$S(g) = s(g)/s_{max} \tag{3.3}$$

The graph s_{max} has by definition $S(g) = 1.0$. It can be thought of both as the most likely graph and also (uniquely) as the most "perfectly" scale-free graph with degree sequence D. Although $s(g)$ and $S(g)$ can be computed for any graph and do not depend on any particular construction mechanism, they have a special meaning in the context of ensembles of graphs. Specifically, $S(g)$ has a direct interpretation as the *relative log-likelihood* of a graph resulting from the GRG construction [24] and thus all of the SF model generation mechanisms generate essentially only high S graphs. The S-metric also potentially unifies other aspects of SF graphs, as it is closely related to betweeness, degree correlation, and graph assortativity, and captures several notions of self-similarity related to graph trimming, coarse-graining, and random rewiring [5]. The focus on ensemble-based methods means that the analysis in SF models has implicitly ignored those graphs that are unlikely to result from such constructions, in particular graphs with small S. Of course the enormity of the number of different high S graphs means that any particular one graph, even the relatively most likely, is unlikely in absolute terms to be selected.

If the toy HOT network is considered (section 3.2.3), the expected result is a high value of S, since it exhibits SF behavior. While $s(g)$ calculation is quite simple, that it is not true for S which require a complex algorithm ([5], Appendix A). However, considering the type of graph (a tree) of this HOT model, it can be greatly simplified. Intuitively, consider a degrees sequence $D = \{d_1, d_2, \ldots, d_n\}$ that can be assumed ordered $d_1 \geq d_2 \geq \ldots \geq d_n$, where d_1 is root degree; then the first d_1 degrees are first level children, for every second level children there will be d_i third level children, etc. This procedure leads to a tree in which the degree is maximum in the proximity of the root and descending to the leaves, which generates a graph as much as possible concentrate where high grades are more connected to others high degrees so then the value of $s(g)$ is maximized ([5], Appendix A.3).

Chapter 4

Implementation

The implementation of the HOT network model in NetLogo (version 5.0.2) uses the set-oriented approach to agents of the tool and some mathematical properties to simplify the algorithms necessary for the generation of the network and computation of the S-metric.

4.1 Generative Algorithm

As pointed out in section 3.2.3 the algorithm try to minimize the cost of two objectives: the Euclidean distance and the number of hops from a central node, which is the root of the tree. Due to the particular structure of the network, if we assume for h_j calculation (c), it is sufficient to consider the level at which a new node is placed in the tree to find the number of hops from the root; it is then defined the turtles variable `nhop`, without resort to algorithms for the calculation of the minimum path, needed in more general graphs.

The algorithm starts with the creation of two nodes (4.1): the first, that is the root, has `nhop = 0`, while the second is placed in a random position within the space, then the two nodes are connected. The second node has `nhop = 1` and is therefore the first child, but you can think about starting a network that consists of several nodes at level 0, without substantially modifying the behavior of the model.

Listing 4.1: Root with `nhop = 0` and a randomly placed node

```
crt 1
[
    set color green
    set nhop 0
]
crt 1
[
```

```
    setxy random-xcor random-ycor
    set color red
    create-link-with turtle 0 [ set color green ]
    set nhop 1
]
```

The next step is repetition of the iterative process of generating new nodes (4.2), which is to produce two random coordinates, and then check which node satisfies the requirements of 3.1; this node is then connected with a child having number of hops incremented by 1 respect to its. The calculation of the Euclidean distance is normalized, since Fabrikant in [2] refers to the Euclidean distance in the *unit square* so it is necessary that $x, y \in [0, 1]$. Coordinates normalization is done through a technique called *Min-Max Normalization*, which transform a value x in x_{norm} so that $x_{norm} \in [a, b]$:

$$x_{norm} = \frac{x - x_{min}}{x_{max} - x_{min}}(b - a) + a$$

Given two points $A \equiv (x_0, y_0)$ and $B \equiv (x_1, y_1)$, replacing in the Euclidean distance $\sqrt{(x_1 - x_0)^2 + (y_1 - y_0)^2}$ normalized coordinates for interval $[0, 1]$ led to:

$$\sqrt{\left(\frac{x_1 - x_{min}}{x_{max} - x_{min}} - \frac{x_0 - x_{min}}{x_{max} - x_{min}}\right)^2 + \left(\frac{y_1 - y_{min}}{y_{max} - y_{min}} - \frac{y_0 - y_{min}}{y_{max} - y_{min}}\right)^2}$$

Finally, simplifying fractions with a common denominator:

$$\sqrt{\left(\frac{x_1 - x_0}{x_{max} - x_{min}}\right)^2 + \left(\frac{y_1 - y_0}{y_{max} - y_{min}}\right)^2}$$

In this case (x_1, y_1) are the randomly generated coordinates, while (x_0, y_0) are coordinates of the node from which you want to measure the distance. The weight `alpha` is implemented by using a global variable associated with a slider in the user interface of the simulation in order to allow a direct control over its value; given the constraints of the model its value is bounded in the interval [4.100], enough to cover graphs of more than 10,000 nodes. To make the model independent from the size of the space (i.e., the number of patches) the model uses variables `max/min-pxcor` and `max/min-pycor`.

Listing 4.2: New nodes must satisfy HOT objectives

```
to go
  ask links [ set color gray ]

  let x random-xcor
  let y random-ycor
```

```
6     let partner nobody

      set partner min-one-of turtles
9     [
        alfa *
        sqrt
12      (
          ( (x - xcor) / (max-pxcor - min-pxcor) ) ^ 2 +
          ( (y - ycor) / (max-pycor - min-pycor) ) ^ 2
15      )
        + nhop
      ]
18    crt 1
      [
        setxy x y
21      set color red
        if partner != nobody
        [
24        create-link-with partner [ set color green ]
          set nhop 1 + [ nhop ] of partner
        ]
27    ]

      tick
30  end
```

4.2 S-metric Computation

The Reporter `relative-log-likelihood` is a procedure which returns $S(g) = s/s_{max}$ for each tick of the simulation, that is, the value of S is recalculated each time a node is added.
Computation of S(g) happens in two steps:

1. calculation of s (4.3): it is requested to the set of links to calculate the quantity $d_i d_j$, that is the weight in 3.2 and then sum it to `s`. This means that each link must read the degree of its vertices and then calculate the required amount to sum to the variable `s`; when every link has done it, `s` will contain the value of s

2. calculation of s_{max} (4.4): it calculates the degree sequence D of nodes of the graph and sort it in descending order; then it proceed by iterating through the list just created. For each step $d_i d_j$ is summed to `smax`, where `di` is "father" node and `?` is the child node (current item in the list). Since every father has $d_i - 1$ children (root excluded which has exactly d_1 children), `di`

is updated after $d_i - 1$ iteration with his son having the highest degree and that can accept at least another children, that is, d_{i+1}. Degrees ordering guarantees (*Rearrangement Inequality* [25]) maximization of the sum `smax` and so at the end it will have the value s_{max}

It must be emphasized that this process is valid only if the degree sequence D satisfies (in this case by construction) the relation:

$$\sum_i d_i = 2(n-1)$$

where n is the number of nodes in the graph. Then all simple connected graphs having degree sequence D correspond to trees so this procedure ensures that you get the graph s_{max}.

Listing 4.3: Algorithm to calculate $s(g)$

```
let s 0
ask links
[
  set s s + [ count link-neighbors ] of end1 *
            [ count link-neighbors ] of end2
]
report s
```

Listing 4.4: Algorithm to calculate s_{max} for tree graphs

```
to-report relative-log-likelihood
  let smax 0
  let counter 0
  let di 0
  let child 0

  let degree-sequence sort-by > [ count  link-neighbors ] of turtles
  set di item 0 degree-sequence
  set degree-sequence remove-item 0 degree-sequence
  foreach degree-sequence
  [
    set smax smax + di * ?
    set counter counter + 1
    if di = counter
    [
      set counter 1
      set di item child degree-sequence
      set child child + 1
    ]
  ]
  report log-likelihood / smax
end
```

Chapter 5

Test and Results

This chapter shows results obtained from a series of simulations performed with the model created. First part consist in the analysis of the properties and topology of HOT networks, while varying of the parameter α and then comparing them with the SF network model based on Preferential Attachment (PA). Second part presents a possible extension of the model.

5.1 HOT vs Preferential Attachment

Fabrikant's model does not differ substantially from that Preferential Attachment as regards the properties of the network, in fact, the central node turns out to be, in almost all cases, the highest degree node and the nodes attached to it appear to be those immediately successive in the degrees sequence. Moreover, scale-free appearance is evident and perhaps even more pronounced than in networks generated by PA (fig. 5.1). In both cases, models have a power-law distribution, but that of the HOT model is "sharper" which is a sign of the fact that the number of hubs is smaller; the distribution tends to round up when α grows, as this parameter controls the "concentration" of the network. HOT model exhibits an evident *exponential cutoff* when α approaches the value $\sqrt{n}$, sign that the tree tends to take on the features of Euclidean minimum spanning tree. Finally it can be seen as the slope of the "straight" on $log - log$ diagrams is less than that of Barabási model, which predict an exponent $\gamma = 3$ (fig. 5.2, 5.3, 5.4, 5.5). From these early observations it is plausible to expect very high values of S, even in comparison to Barabási's model. However, while the calculation of s_{max} for HOT network is possible, Preferential Attachment involve the use of the full algorithm, hardly implementable in NetLogo, so the solution is to compare s. Note that s is directly comparable if and only if the graphs have the same degrees sequence D. An alternative approach is to generate a large number of configurations, and then

analyze in what range varies $s(g)$.

Results generated by 1000 configurations of 3000 nodes each (Table 5.1) suggest some interesting observations. The PA Model generates graphs with a value of $s(g)$ on the "average", but with a discrete variability, in fact the range of values covered by s is quite large and the standard deviation is therefore high compared to the mean value (Fig. 5.8), this indicates that in the process of preferential attachment they are possible, though unlikely, highly concentrated or distributed configurations, due to its stochastic nature.

	PA	HOT $\alpha = 4$	HOT $\alpha = 20$	HOT $\alpha = 50$
AVERAGE	133491	2280752	185789	71596
STDEV	28940	67210	7997	2001
MIN	78697	2043928	158312	63704
MAX	317173	2492921	207439	78604

Table 5.1: Preferential Attachment (PA) vs. HOT: statistical characterization of $s(g)$ on varying of α.

The behavior of HOT model is instead strongly influenced by the value of α with regard to the order of magnitude of $s(g)$. For little values, you get a graph very concentrated where the root node is significantly "larger" (i.e., with more links) of the others (Fig. 5.6). Furthermore, s is found to have a limited dispersion, then the model has a low variability, that tends to generate similar values of s or, seen from another perspective, we can say that *by construction*, due to the constraints to which the model must undergo, it generates a value of s in a narrow range. In particular, for $\alpha = 4$ the model generates a graph with $s \approx s_{max}$, in agreement with what have been shown by Fabrikant. With the increase of α the network tends to become much more "distributed" (Fig. 5.7), so much to lose almost hubs, while the value of s decreases one or two orders in magnitude (Fig. 5.9). However, the range of values of $s(g)$ remains small, in fact the relationship between the mean value and standard deviation is nearly constant, according to the expectations, since constraints were not changed, but just their relative weight.
What have been shown for s is reflected in the value assumed by $S(g)$ (Table 5.2). For $\alpha = 4$ the model gets to generate a configuration having $s = s_{max}$, again in agreement with what was said by Fabrikant, and always maintains very high values of the metric S. In the case of $\alpha = 20$ or $\alpha = 50$ it seems to saturate at an average value of $\overline{S} \simeq 0,815$ with upper bound $S_{max} \simeq 0.90$ and lower bound $S_{min} \simeq 0.70$, regardless of the value of α. Although it is not really clear why, it is logical to assume that the model itself, tied to a tree topology, is responsible for the fact that s can not fall below a certain value, to put it another way, links between the nodes of the graph are bound to form a tree and then even nodes degrees.

$S = s/s_{max}$	$\alpha = 4$	$\alpha = 20$	$\alpha = 50$
AVERAGE	0.987554152	0.8122539989	0.8170608161
STDEV	0.009035764	0.0241575629	0.0130547917
MIN	0.945879733	0.7217450373	0.7780226344
MAX	1.0	0.8943435835	0.8574272853

Table 5.2: Relative Log Likelihood on varying of α.

If we compare these values of S with those obtained by Doyle in [4], we note that the HOT model presented here, as indeed expected, is part of the SF models for "performance" according to the metric $P(g)$, which consists in the maximum throughput of the network under a "gravity model" for the end user traffic demand. In this sense, this model does not realistically represent Internet, where the performance is rather a crucial aspect. The dissimilarities with Doyle's model do not stop here:

- in Fabrikant's model all nodes are *peers*, while Internet exhibits layers and hierarchies both from the point of view of connectivity and from the nodes themselves
- a node can have an arbitrary number of links, while in Internet since the cost of maintaining links increases with their number, as you move up in the hierarchy you employ technology to aggregate traffic on the same physical link (e.g. Wavelength Division Multiplexing)
- hub nodes are in the center of the network, while in Doyle's model at the outskirts
- resulting network is scale-free instead of scale-rich

You can not assume, however, that it is a wrong model, since all of these features may possibly be achieved by adding additional constraints and properties that consider technological aspects of telecommunications networks. It can be said that the Fabrikant's model, in its simplicity and generality, offers a universal tool for HOT modeling.

5.2 Towards a Structural-Generative Model

The structuralist approach of Doyle's model has the limitation of being bound to the topology choice for that application field and requires specific adaptations; also assumptions are made, such as the choice of a core of only four nodes connected by a fully connected mesh, that they are hardly justifiable within a more extensive

growth of the core of the network, so it can't be considered actually a general model. Another limitation is the inability to generate power-law distribution by itself, but it is based on building from a given degree sequence D. This makes the model completely unable to give prediction indeed it is limited to a mere post-hoc analysis, so it is useless to solve a problem of heuristically optimal topology outside of its scope.

In an attempt to create a more realistic generative model it is possible to emulate part of the topology shown by Doyle changing the `setup` method (5.1) in order to create an arbitrary number of initial nodes, placed in a random position in the space and each one connected to two other nodes. The result is a network with the same power-law in node connectivity (fig. 5.10), but calculation of $S(g)$ is no longer valid. If you try to see how varied the value of s in the set of 1000 graphs of 3000 nodes each with $\alpha = 20$, we find that $\overline{s} \simeq 200000$, therefore comparable to the original model, since the core nodes are those with highest degree; however, we expect more resilience to attacks and failures due to redundant links, then adds one of the features of Doyle's HOT model.

Listing 5.1: Create some interconnected nodes with `nhop = 0`

```
to setup
  clear-all
  set-default-shape turtles "circle"

  repeat 10
  [
    crt 1
    [
      setxy random-xcor random-ycor
      set nhop 0
    ]
  ]
  ask turtles [ create-links-with other n-of 2 turtles ]

  reset-ticks
end
```

This result leads to a further observation: in Doyle's HOT network "core" nodes are basically disconnected from "user" nodes, thanks to the interposition of "edge" nodes (e.g. Border Router). You can then create a network whose initial nodes are "core" with `nhop = 0`, as showed before, and some "edge" nodes with `nhop = 1`, randomly placed in the space and each one connected to a core node; new nodes will be connected only to nodes having `nhop` $\geq$ `1` (5.2). Again resulting network is power-law (fig. 5.11) but this time it has $\overline{s} \simeq 140000$ which is quite less than the one of basic model. Another feature that this model gain is *self-dissimilarity* or *scale-richness*, because characteristics of connectivity between core and edge/user

nodes is different so network appearance is not independent from the scale of observation. Note that the assumptions made about the connectivity and the number of core and edge nodes are completely arbitrary and devoid of theoretical foundations, despite this, the model accomplishes desired goals, indicating a certain generality and flexibility of this structure. It is therefore clear that in order to fully exploit the predictive potential of this generative model, it is necessary to identify the relationships between these quantities and demonstrate rigorously as they lead to a power-law distribution and minimize the Relative Log Likelihood. In the absence of such results, the only statement that can be said is that HOT networks characteristics are exhibited entirety only from hierarchical networks with at least three distinct decoupled levels, because they provide the opportunity to create scale-rich networks.

Listing 5.2: Creation of "core" and "edge" nodes

```
to setup
  clear-all
  set-default-shape turtles "circle"
  repeat 10
  [
    crt 1
    [
      setxy random-xcor random-ycor
      set color green
      set nhop 0
    ]
  ]
  ask turtles [ create-links-with other n-of 2 turtles ]
  repeat 30
  [
    crt 1
    [
      setxy random-xcor random-ycor
      set color red
      create-links-with other n-of 1 turtles with [ nhop = 0 ] [ set
          color green ]
      set nhop 1
    ]
  ]
  reset-ticks
end
```

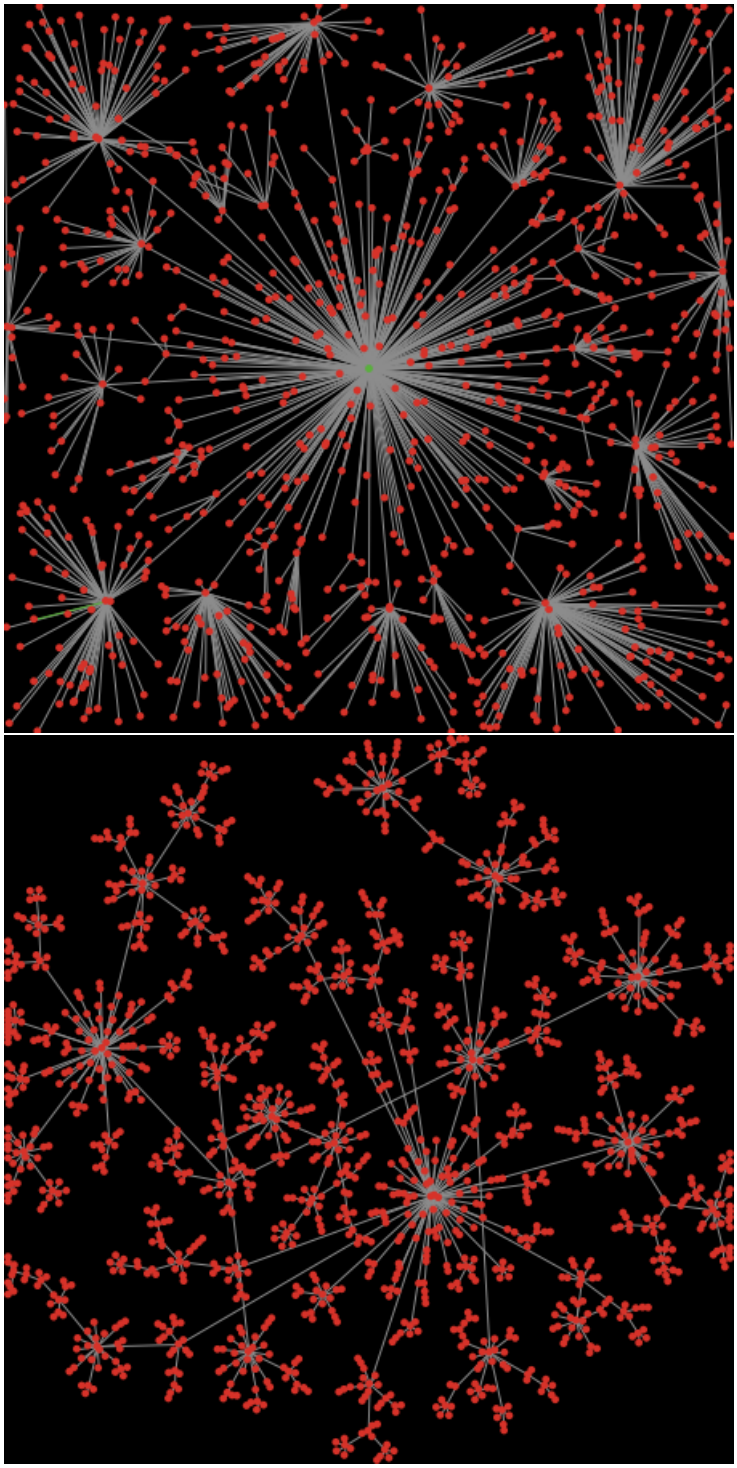

Figure 5.1: Graph with $n = 1000$: HOT ($\alpha = 4$) vs PA.

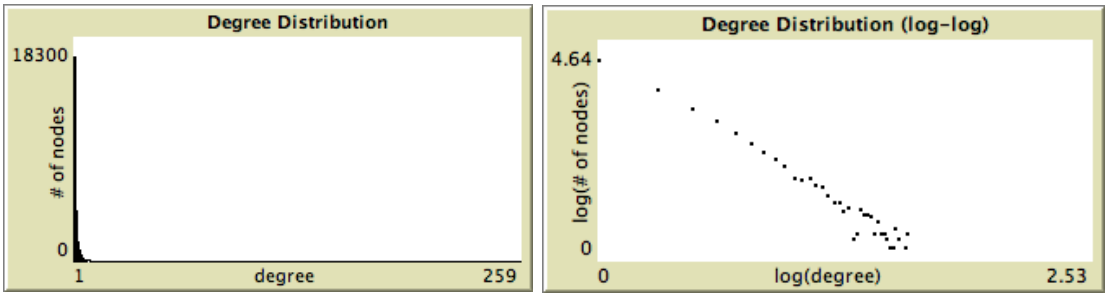

Figure 5.2: Degrees Distribution of PA graph with $n = 25000$.

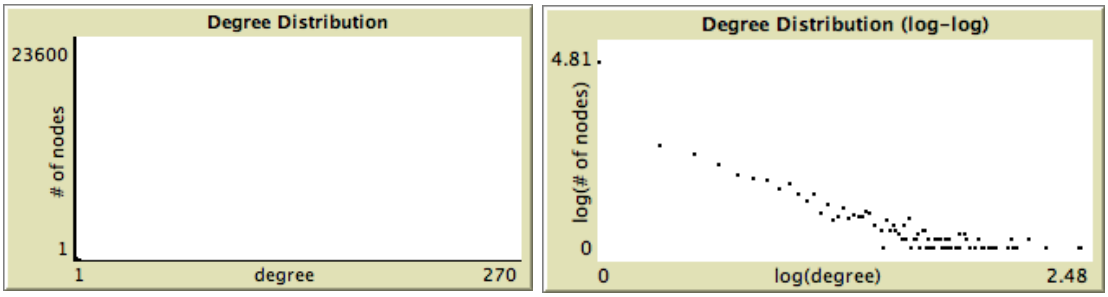

Figure 5.3: Degrees Distribution of HOT graph with $n = 25000$ and $\alpha = 20$.

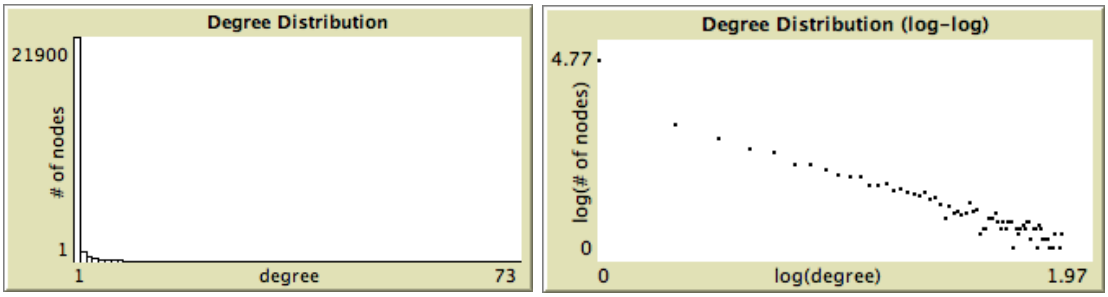

Figure 5.4: Degrees Distribution of HOT graph with $n = 25000$ and $\alpha = 50$.

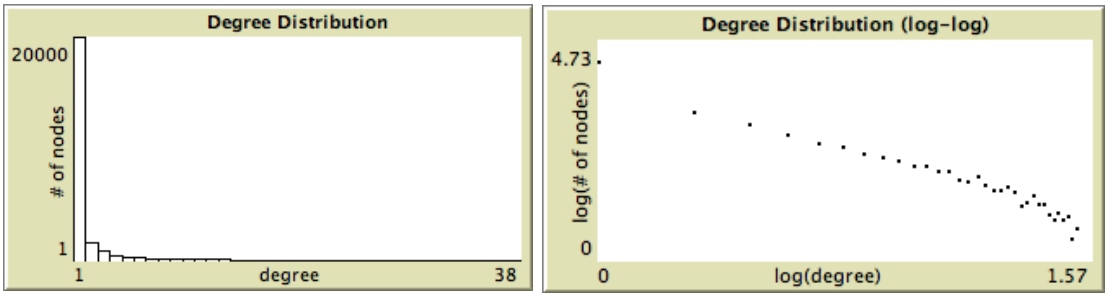

Figure 5.5: Degrees Distribution of HOT graph with $n = 25000$ and $\alpha = 100$.

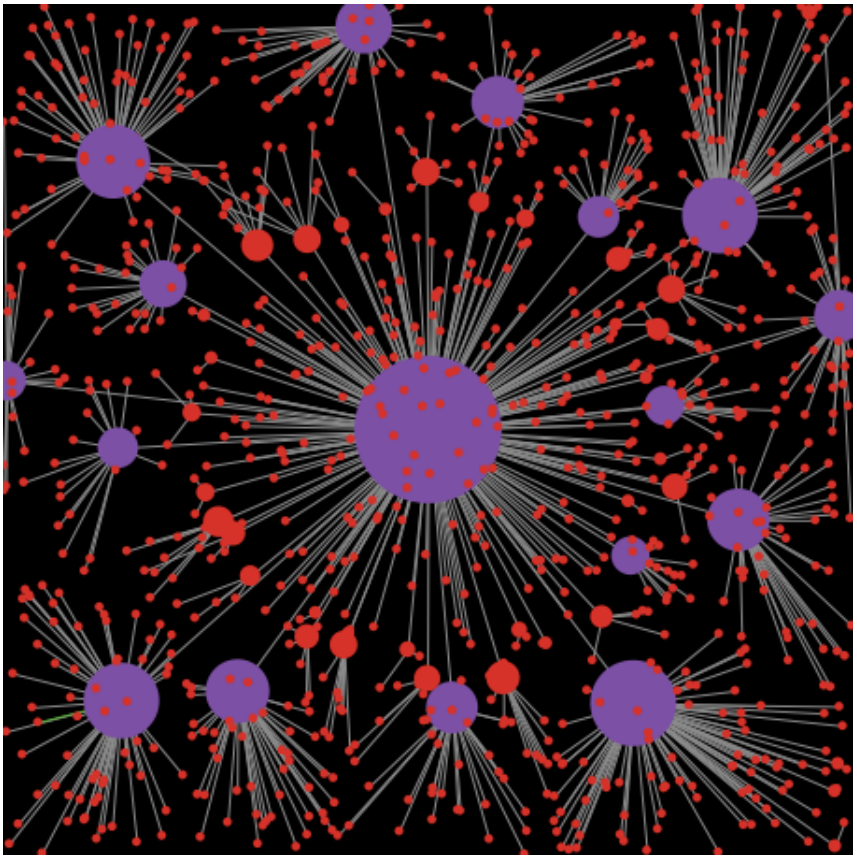

Figure 5.6: HOT Graph with $n = 1000$ $\alpha = 4$. The nodes in purple have more than 16 links.

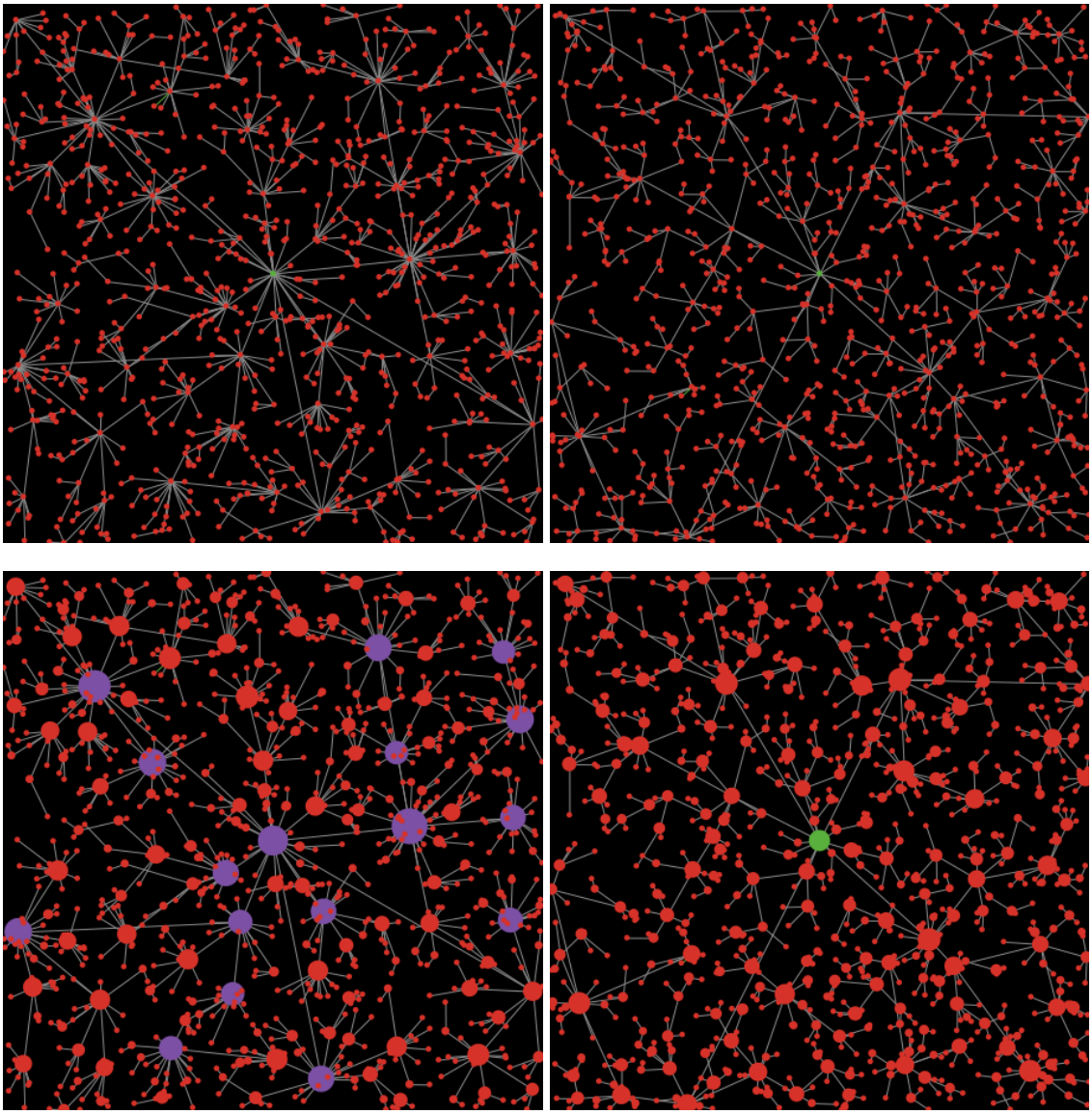

Figure 5.7: HOT Graph with $n = 1000$ and $\alpha = 20$ (left) or $\alpha = 50$ (right). The nodes in purple have more than 16 links.

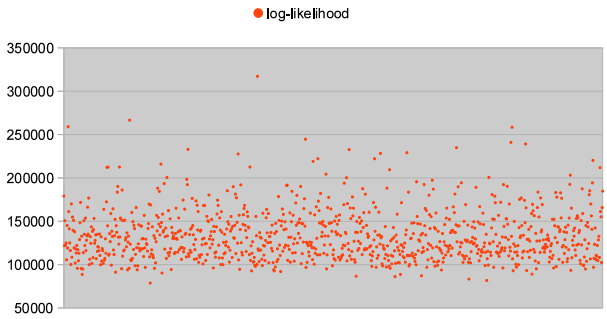

Figure 5.8: Distribution of $s(g)$ values in PA.

Figure 5.9: Distribution of $s(g)$ values in HOT, $\alpha = 4, 20, 50$.

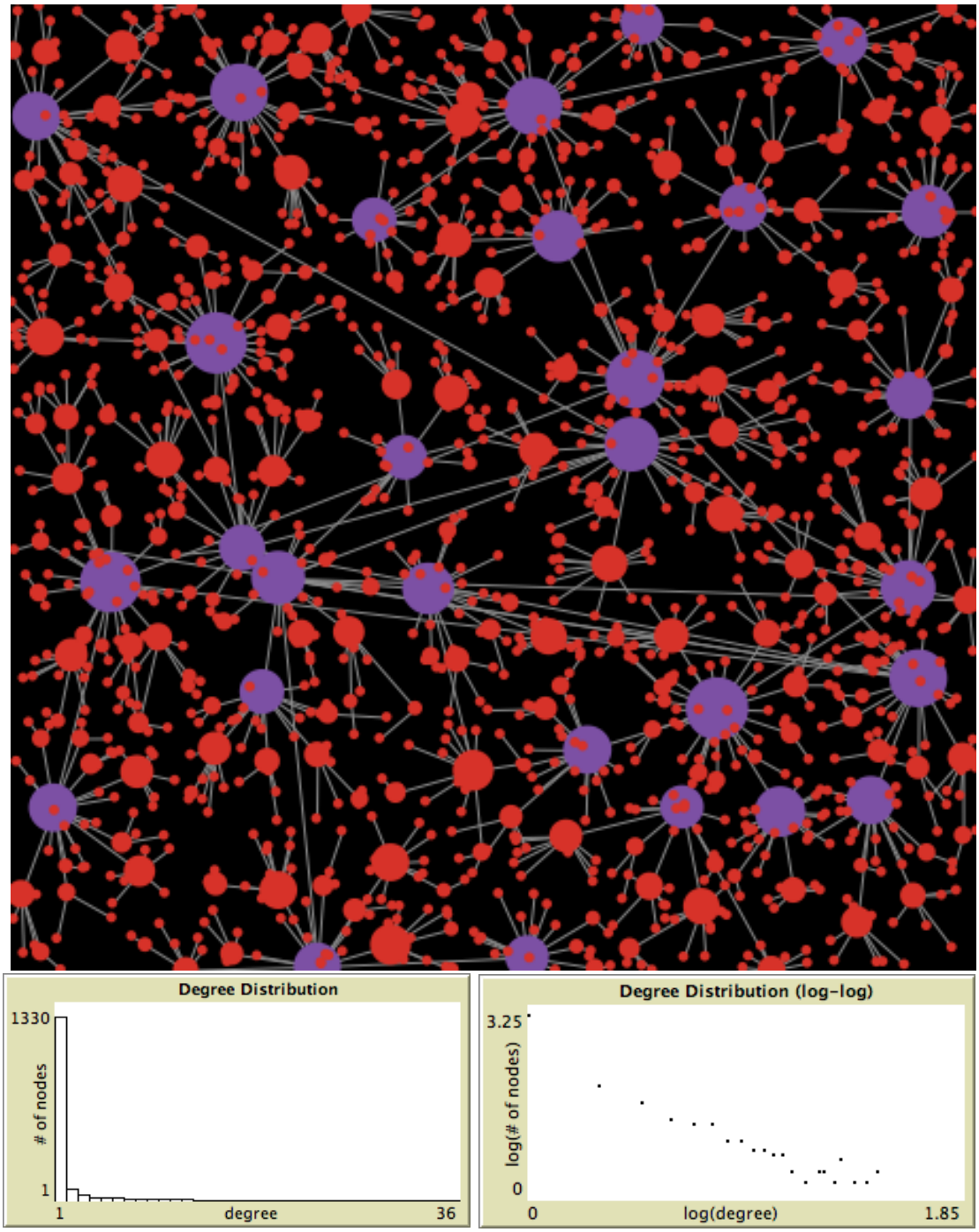

Figure 5.10: HOT Graph with $n = 1500$, $\alpha = 20$, 10 nodes with `nhop = 0`.

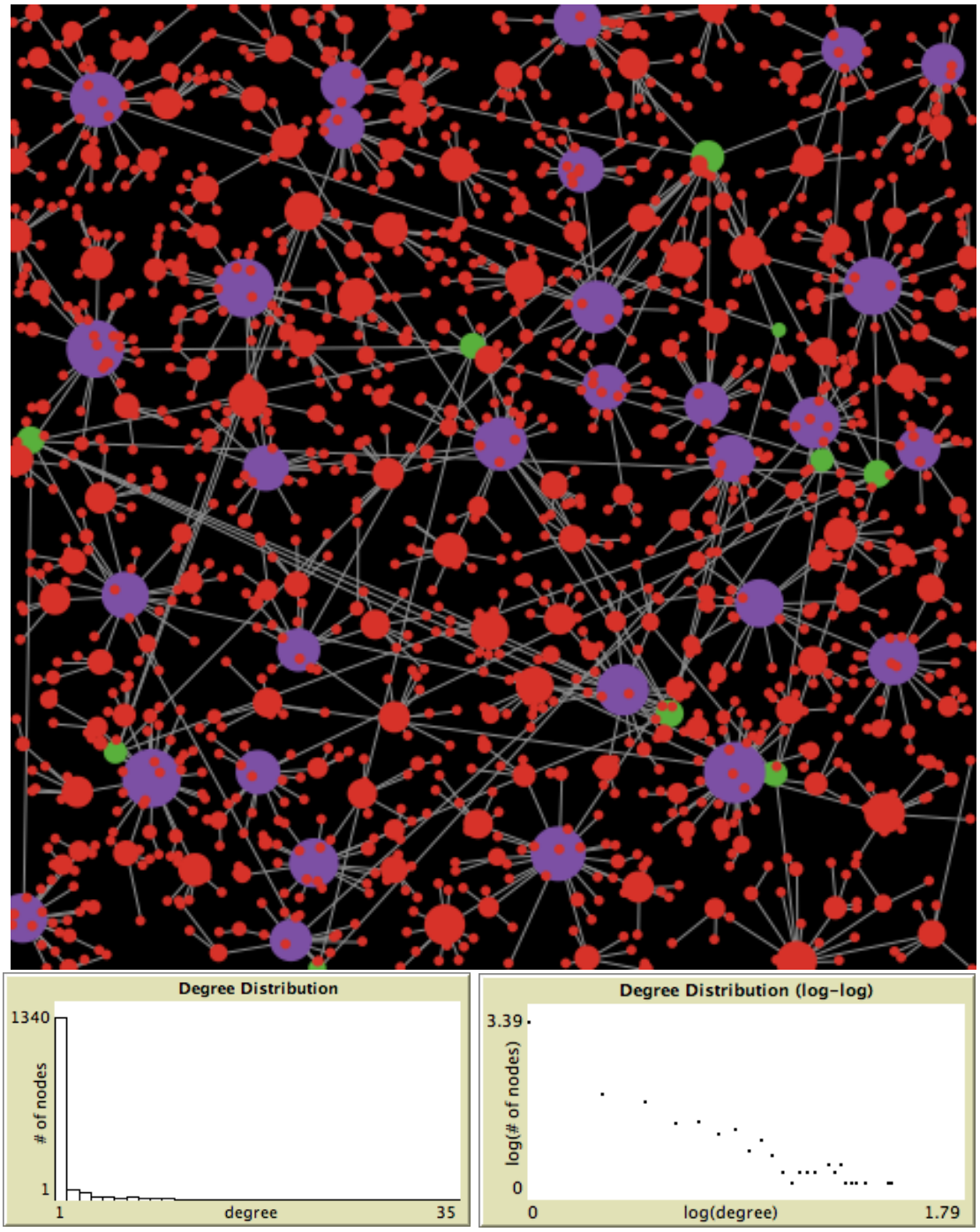

Figure 5.11: HOT Graph with $n = 1500$, $\alpha = 20$, 10 "core" nodes and 30 "edge" nodes. Core nodes are shown in green.

Chapter 6

Conclusions and Open Issues

This chapter present the main and the most important results achieved by the project carried out, comparing them with the initial expectations and offering observations and considerations in this regard. In addition, future developments of the project.

6.1 Results

In conclusion it can be stated that the implemented system represents in satisfactory manner the HOT networks, as it is able to illustrate the principles behind them and produce the expected results in a totally general manner. It has been observed that the power-law distribution emerges from compromises between performance and constraints that need to be optimized simultaneously; the model also exhibits, such as the preferential-attachment of Barabási, a scale-free structure and a behavior such as "the rich get richer", yet this is not due to a primitive assumption, but the results of the local optimization. It is also noted that compared to the model of Doyle has the advantage of having a predictive power and appears to be universal, but at the cost of being limited and unrealistic since it can only represent some of the characteristics of HOT networks. In testing some of the limitations of the model have been addressed and has been proposed a possible evolution that combines the advantages of generative models with those of structured models, making it potentially more effective and powerful than both, because it is able to satisfy a greater number of properties of the Internet graph, beyond the simple power-law distribution, while maintaining the prediction ability.

6.2 Open Issues and Future Development

At present the implementation, although fully functional, it is not applicable to general graphs, but it is bounded to trees. Moreover no distinction between core, edge and user nodes is done, that is, it does not represent the HOT hierarchy of sub-networks. In line with the proposed extension it is possible to highlight some future developments:

- study of the relationships between core, edge and user nodes to allow the implementation of an algorithm, formally valid, in the structural-generative model
- implementation of $S(g)$ computation for general graphs
- implementation of the calculation of other quantities characterizing the properties of the graph

The realization of the first point, that it is also the most important, could potentially be disruptive, because the hierarchical structures and the aggregation of flows by levels are present in a large number of cases of real life, think of as Internet, the corporate structure and social networks, as well as the food pyramid or the cellular structure and anatomy of the organs of living beings.

Appendix A

Run Simulations

To run a simulation simply click on `setup` and then `go`. The switch for the graphics and check to update the model view can be turned off to speed up the simulation. In order to perform more simulations consecutively and save the results to a csv file you can select *BehaviourSpace* from tools menu where you can access a preconfigured experiment.

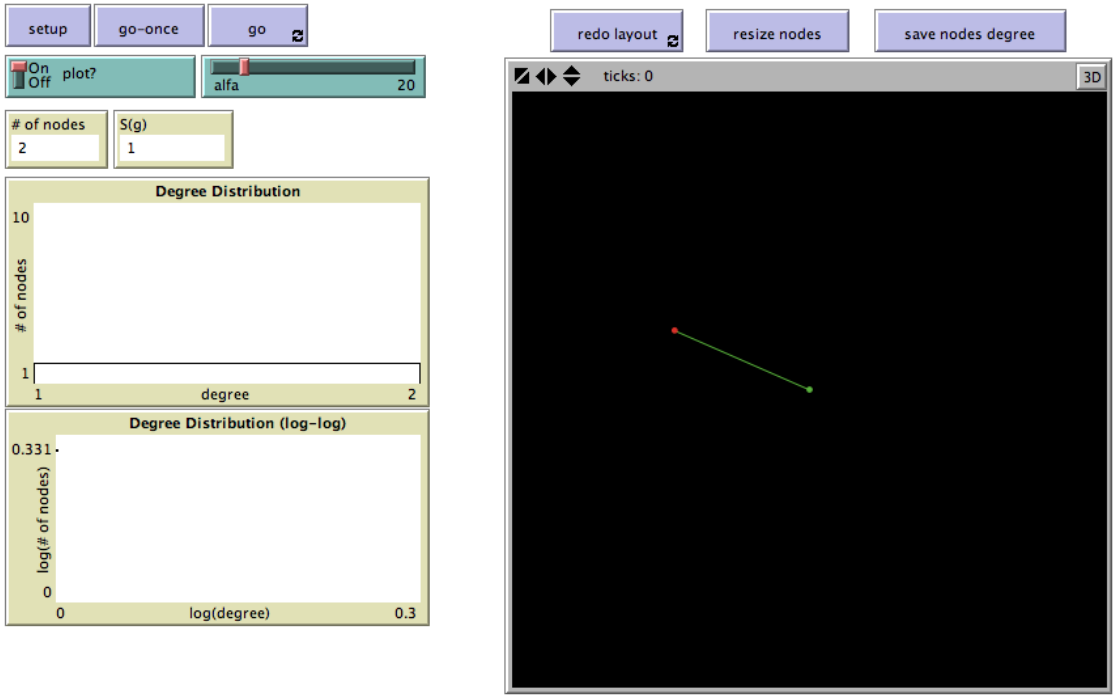

Figure A.1: It is possible to run simulations through a simple interface.

Bibliography

[1] A.-L. Barabàsi and R. Albert, *Emergence of scaling in random networks*, Science 286, 509-512 (1999).

[2] Alex Fabrikant, Elias Koutsoupias, and Christos H. Papadimitriou, *Heuristically Optimized Trade-offs: A New Paradigm for Power Laws in the Internet*, Proceedings of ICALP 2002.

[3] J. M. Carlson and J. C. Doyle, *Highly Optimized Tolerance: a mechanism for power laws in designed systems*, Physics Review E 1999.

[4] J.C. Doyle, D. Alderson, L. Li, S. Low, M. Roughan, S. Shalunov, R. Tanaka, and W. Willinger, *The Robust Yet Fragile Nature of the Internet*, Proc. Natl. Acad. Sci. USA 102(41). 2005.

[5] L. Li, D. Alderson, J.C. Doyle, and W. Willinger, Towards a Theory of Scale-Free Graphs: Definition, Properties, and Implications, Internet Math (2005).

[6] NetLogo: a multi-agent programmable modeling environment; http://ccl.northwestern.edu/netlogo/.

[7] V. Pareto, *Cours dEconomie Politique*, Dronz, Geneva Switzerland, 1896.

[8] G. Zipf, *Human behavior and the principle of least effort*, Addison-Wesley, Cambridge MA, 1949.

[9] X. Gabaix, *Zipfs law for cities: an explanation*, Quarterly Journal of Economics, 114:739767, 1999.

[10] B. Mandlebrot, *An informational theory of the statistical structure of languages*, in W. Jackson, editor, Communication theory, pages 486502. Betterworth, 1953.

[11] R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins, *Extracting large scale knowledge bases from the Web*, in Proceedings of the 25th VLDB Conference, 1999.

[12] Mandelbrot, B.B., *Fractals and Scaling in Finance*, Springer-Verlag, New York, 1997.

[13] M. Mitzenmacher, *A Brief History of Generative Models for Power Law and Lognormal Distributions*, Internet Mathematics, vol 1, No. 2, pp. 226-251, 2004.

[14] NetLogo Model Library:
http://ccl.northwestern.edu/netlogo/models/index.cgi.

[15] NetLogo Community's Models:
http://ccl.northwestern.edu/netlogo/models/community/index.cgi.

[16] NetLogo Programming Guide:
http://ccl.northwestern.edu/netlogo/docs/programming.html.

[17] NetLogo Dictionary:
http://ccl.northwestern.edu/netlogo/docs/dictionary.html.

[18] Clark, D.D., *Proc. ACM SIGCOMM88*, in ACM Comp. Comm. Rev., 18(4): 106114.

[19] W. Willinger, J.C. Doyle, *Robustness and the Internet: Design and Evolution*, in "Robust design: A Repertoire of Biological, Ecological, and Engineering Case Studies", E. Jen, Editor, Oxford University Press, 2004

[20] D. Bank, R. Richmond, *Where the dangers are*, The Wall Street Journal, July 18, 2005; page R1.

[21] C. Faloutsos, M. Faloutsos, P. Faloutsos, *On power-law relationships of the internet topology*, in Proc. SIGCOMM, 1999.

[22] H. Tangmunarunkit, R. Govindan, S. Jamin, S. Shenker, W. Willinger, *Network topologies, power laws, and hierarchy*, Technical Report 01-746, Computer Science Department, University of Southern Calofornia, 2001.

[23] Hongsuda Tangmunarunkit, Ramesh Govindan, Sugih Jamin, Scott Shenker, Walter Willinger, *Network Topology Generators: Degree-Based vs Structural*, Sigcomm, 2002

[24] F. Chung, L. Lu, *Internet Math.*, 91113, (2003).

[25] K.Wu and A. Liu, *The Rearrangement Inequality*:
https://umdrive.memphis.edu/ccrousse/public/PUTNAM/Rear.pdf.