

# NAT (Network Analysis Tools) Suite

## User's Manual

## Part I

# Introduction

**NAT** is a web tool for the analysis of topological parameters of a graph. The tool has been conceived and realized to apply ideas and methods of *Complexity Science* in the area of Large Complex Critical Infrastructures (LCCI) analysis and protection. It has been realized by ENEA in the frame of the EU-funded project "IRRIIS" (Integrated Risk Reduction of Information-based Infrastructure Systems, IST Project N. 027568). The technical realization of the tool has been committed by ENEA to Ylichron Srl through the Contract prot. ENEA/2006/49682/FIM-INFO-AFU.

Each LCCI can be ultimately represented by a **network** containing the logical position of its different constitutive elements, connected through (logical or physical) links. Networks can be subsequently mapped onto **graphs** which constitute the ultimate, and more abstract, layout of a technological infrastructure, where only the mathematical structure of its constitutive elements and their connections are kept.

Graphs can be thus analyzed by methods typical of the Graph's Theory; the final goal is to infer the estimate of relevant network's properties from the analysis of the associated graph and thus to gain insights on the effective properties of the "real" technological object represented by the network.

A graph can be described as the set  $G = G(N, L)$  of  $N$  nodes and  $L$  arcs (or links). A graph can be mathematically defined by its *Adjacency* matrix **A** whose elements will be indicated as  $a_{ij}$ .

The "conceptual" layout of this analysis is thus the following:

- the physical (or logical) map of the infrastructure should be transformed into a network
- the network is transformed into an Adjacency matrix
- the Adjacency matrix is elaborated to provide the estimate of the topological properties.

## Part II

# Networks Builder

## 1 Types of Complex Networks

The **Networks Builder** tool of the **NAT** suite enables the user to generate different kinds of complex networks, distinguished by their topological properties and thus generated with different growth mechanism. Networks are represented by graphs, particularly by *undirected* and *unweighted* graphs. An undirected graph is a graph for which the relations between pairs of nodes are symmetric, so that each edge has no directional character (as opposed to a directed graph). This means that, if an edge between the nodes  $i$  and  $j$  (represented by  $(i, j)$ ) exists, the reverse edge  $(j, i)$  also exists. An unweighted graph instead, is a graph in which all edges have no label and the same unitary weight.

The network's topologies created by the **Networks Builder** tool are the followings:

- **Random Graphs**

A random graph is a graph that is generated by some random process. In the **NAT Networks Builder** the used growth mechanism is the **Erdős-Rényi**<sup>1</sup> one. The growing mechanism for the creation of a random graph  $G = (N, L)$  starts with a set of  $N$  nodes with no connections. The process continues iteratively adding new nodes at each iteration according to an unconditional probability  $P$ . It means that the edge creation probability between two nodes, the new node  $i$  and the pre-existing node  $j$  ( $P(i \rightarrow j)$ ), is constant and node independent:

$$P(i \rightarrow j) = \frac{1}{N} \quad (1)$$

This leads to a typical nodes degree distribution, best fitted by a poissonian function; therefore, the probability for a node to obtain  $k$  links is:

$$P(k) = C_{N-1}^k p^k (1-p)^{N-1-k} \quad (2)$$

and for  $N \rightarrow \infty$ :

$$P(k) = e^{-\langle k \rangle} \frac{\langle k \rangle^k}{k!} \quad (3)$$

as shown in figure 1.

---

<sup>1</sup>P. Erdos, A. Rényi, Publ. Math. Debrecen 6 (1959) 290.

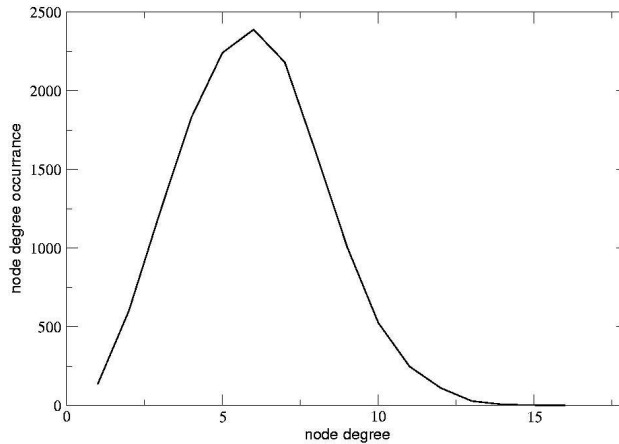


Figure 1: *Nodes' degree distribution in a random graph,  $N = 14154$ .*

- **Scale Free Networks**

Recent studies about “real networks” in several domains, from biology to sociology to telecommunications, have shown that the homogenous nodes' degree distribution typical of random graphs isn't the best way to represent these systems. Nodes' degree distribution of these systems seems to follow a *Power Law* function:

$$P(k) \sim k^{-\gamma} \quad (4)$$

in which  $2 < \gamma < 3$  according to singles systems' peculiarities. These kinds of networks are defined **Scale-Free**, because *Power Law* is the unique functional form that remains unchanged, at less of a multiplicative factor, after a scale modification.

**Scale Free** networks are characterized by an extremely inhomogeneous nodes degree distribution, with the presence of few nodes highly connected (known as *hubs*) and a lots of loosely connected ones (known as *leaves*).

To reproduce scale free networks, NAT's **Networks Builder** tool uses the **Barabási-Albert**<sup>2</sup> (BA) growing model. This model is based on two elements: the growing process and the *Preferential Attachment* (PA). The model is inspired by the “rich get richer” principle, in which highly connected elements have an higher probability to attract new connections.

---

<sup>2</sup>A.L. Barabási, R. Albert, Science 286 (1999) 509.

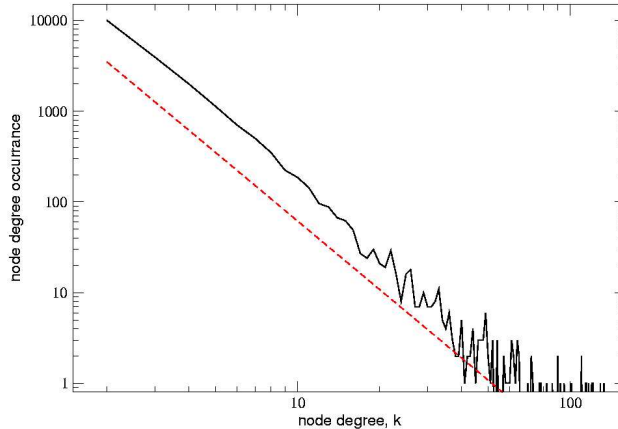


Figure 2: *Nodes degree distribution of a scale free network,  $N = 20000$  (continuous line) the curve of equation  $y = k^{-2.51}$  (dashed line)*

The growing mechanism for the creation of a  $N$  nodes network prescribes an initial phase, in which a set of  $m_0$  ( $m_0 < N$ ) interconnected nodes is created and a growing phase divided into  $N - m_0$  time steps. At each time step a new node is added to the network and connected through  $m_0$  new edges. The choice of the pre-existing nodes for the creation of new edges is made using the *preferential attachment* mechanism. The PA prescribes that the node's probability to obtain a new link is proportional to its degree. In a formal way, the probability for a newly added node  $i$  to establish a connection with the pre-existing node  $j$  ( $P(i \rightarrow j)$ ) is:

$$P(i \rightarrow j) = \frac{k_j}{\sum_j k_j}, \quad (5)$$

in which  $k_j$  is node  $j$ 's degree. For  $N \rightarrow \infty$  this model leads to a degree distribution which follows a *Power Law* with a  $\gamma = 3$  exponent.

- **Internet-Like Scale-Free model**

The **BA** mechanism can reproduce most of “real” networks but, as each network type has its own peculiarities, to reproduce the relevant properties of different networks several modifications to the growth mechanisms should be added. The Internet network, for instance, has been shown to display an “extreme” Scale-Free topology, with the presence of higher degree hubs, a bigger number of leaves, an extremely

different clustering coefficient with respect to “normal” Scale-Free networks. For better reproducing the Internet topology, a new growth mechanism has been proposed. The mechanism is an evolution of the Barabási-Albert one, based on a generalization of the PA, modified with some new elements. To mimic higher degree hubs typical of the Internet, the PA mechanism has been generalized by introducing an adjustable parameter  $\alpha$ :

$$P(i \rightarrow j) = \frac{k_j^\alpha}{\sum_{j=1}^P k_j^\alpha} \quad (6)$$

if  $\alpha < 1$  the strenght of an hub to attract new links is reduced, while if  $\alpha > 1$  is increased.

To increase the network clustering coefficient, the generalized PA mechanism is reinforced through the *Triad Formation* (TF) mechanism. This prescribes that, once a first link with a node  $i$  is established with the PA, the following links of the new node will be established with one of  $i$ 's neighbors. This leads to increase the number of triangles in the network and, therefore, increase the clustering coefficient. The last modification made to the model pertains to the value of the  $m_0$  parameter. In the BA model, the parameter  $m_0$  prescribes the number of link established at every time step; in the proposed model, to increase the number of leaves, the number of newly added links at every time step is changed according to the followings possibilities:

1. At every time step, an integer value  $m_i$  with  $1 < m_i \leq m_0$  is randomly chosen.
  - if  $m_i = 1$  only a new link will be established in this time step and, for the 3 or 4 (50% probability) upcoming time steps,  $m_1$  will be set to 1.
  - if  $m_i = 3, 4$  or 5 two new links will be established with a probability of 95% or the original value of 3, 4 or 5 links is used. No bonds are imposed for the upcoming time steps.
  - if  $m_i = 2$  or 6 the  $m_i$  value is used to establish new links, and no bonds are imposed for the upcoming time steps.
  - Otherwise, the  $m_i$  chosen value is used.
2. The links are established with a previous bond.

These modifications allow to generate a network structure topologically very similar to the Internet. The complete growth mechanism results thus formed by the following chain of actions:

1. A set of  $m_0$  fully connected nodes is created.
2.  $N - m_0$  iterations are performed, at every iteration:
  - A new value for  $m_i$  is randomly chosen and the number of newly added links is selected.
  - A new node is added to the network.
  - The new node is connected to the network: the first link is established with the modified PA mechanism, the remaining links will be established with a linear combination of PA and TF.

$$G(i, 1) = PA, \quad (7)$$

$$G(i, 2..m_i) = (1 - q)PA + qTF \quad (8)$$

Where  $G(i, j)$  is the mechanism chosen for the creation of the link, and  $q$  is an adjustable parameter of the system (the best value to reproduce the Internet structure is  $q = 0.93$  and  $\alpha = 1.44$ ).

## 2 Usage

**NAT's Networks Builder** tool can be used via registration and authentication at the web site [irriis.nat.ylichron.it](http://irriis.nat.ylichron.it); this page could be also accessed through the project IRRIS homepage [www.irriis.org](http://www.irriis.org).

### 2.1 Input Data

- A valid e-mail address for the results. Once the tool has created the network in a text file (.txt), the file is sent by e-mail to the user.
- Type of network. The type of network that wants to be created. There are three possible types of network:
  1. *Random graphs*: the graph is created according to the **Erdős-Rényi** model.
  2. *Scale-Free networks*: the graph is created according to the **Barabási-Albert** growth mechanism.
  3. *Internet-Like Scale-Free networks*: the graph is created according to the proposed model eqs.(7,8) to better reproduce the Internet structure.

- Number of nodes in the network. Specify the dimensions of the network. The number of edges is defined according to the network's type and the number of nodes.
- The  $m_0$  parameter. This parameter is used only in case of a Scale-Free (**Barabási-Albert** or **Internet-Like**) network is selected.

## 2.2 Output Data

The Output data of the **NAT's Networks Builder** tool consist of a text file (.txt) in which the network's structure is stored. Network structure is represented as follow:

- Each node is represented by an identification number (ID), the numbering starts from 0, so the first node has ID= 0, the second has ID= 1 and so on.
- Each edge is represented by a couple of nodes, therefore, if the edge (1, 5) exists, in the output file will appear a line with 1 and 5. In the tool all the graphs are undirected, this means that the edge (1, 5) and the (5, 1) represent the same link, so only the first edge is represented; therefore, in the output file will appear only the line 1 5 and not the line 5 1.
- Each edge is stored in a single line, the number of edges in the network is equal to the number of lines in the file.

In the following figures an example of an undirected graph and the correspondent output file are shown.

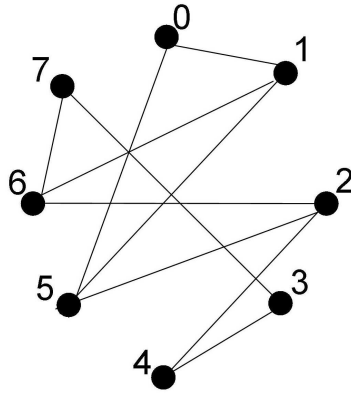


Figure 3: *An example of undirected graph, with 8 nodes and 10 edges.*

0	1
0	5
1	5
1	6
2	4
2	5
2	6
3	4
3	7
6	7

Table 1: *Example of output file structure.*

## Part III

# Topology Analysis of Unweighted Networks

## 1 Topological properties of an unweighted graph

The tool for the topological analysis of unweighted networks of the **NAT** suite enables the user to evaluate different topological properties of a user-defined unweighted (and undirected) graph which can be uploaded on the tool web site from the user repository. An undirected graph is a graph in which every link is bidirectional. The topological properties evaluated by such specific **NAT** tool are the following:

- **Connect**

Verification if the graph is connected or disconnected. A graph is defined as connected if there is a path connecting each pair of nodes  $i$  and  $j$  (with  $i \neq j$ ).

- **Graph degree**

Evaluation of the degree distribution  $P(k)$ , which is the fraction of nodes in the graph displaying degree  $k$ . The degree  $k_i$  of a node  $i$  is the number of edges incident to that node, and can be expressed by means of the *Adjacency* matrix  $\mathbf{A}$  as:

$$k_i = \sum_{j=1}^N a_{ij} \quad (1)$$

where  $N$  is the number of nodes of the graph and  $a_{ij}$  identifies the elements of  $\mathbf{A}$ .

- **Graph Spectrum**

Evaluation of the graph spectrum, corresponding to the set of eigenvalues of its Adjacency matrix  $\mathbf{A}$ . A graph  $G(N, K)$  has  $N$  eigenvalues  $\epsilon_i$  ( $i = 1, 2, \dots, N$ ) and  $N$  associated eigenvectors  $\nu_i$  ( $i = 1, 2, \dots, N$ ). For an undirected graph,  $\mathbf{A}$  is real and symmetric, hence the graph displays real eigenvalues  $\epsilon_1 < \epsilon_2 < \dots < \epsilon_N$ , and the eigenvectors corresponding to distinct eigenvalues are orthogonal. Aside to the *Adjacency* matrix  $\mathbf{A}$ , it can be defined the so-called *Laplacian* matrix  $\mathbf{L}$  which is defined

as:

$$L_{ij} = \begin{cases} \sum_{j=1}^N A_{ij} & \text{if } i = j \\ -A_{ij} & \text{if } i \neq j \end{cases} \quad (2)$$

- **MinCut**

Evaluation of the min-cut, consisting in the optimal partitioning of the graph into two connected subgraphs (of dimensions  $N_1$  and  $N_2$ , such as  $N = N_1 + N_2$ ) separated by  $K$  edges.

- **Cluster**

Evaluation of the clustering coefficient  $C$ , deriving from the average, over all the nodes of the graph, of the node's clustering coefficient  $c_i$ :

$$C = \frac{1}{N} \sum_{i=1}^N c_i = \sum_{j,m \in n_i} \frac{a_{ij} a_{jm} a_{mi}}{n_i(n_i - 1)/2} \quad (3)$$

where  $n_i$  is the number of nearest neighbors of node  $i$ ,  $n_i^*$  is the number of links existing among the nearest neighbors of node  $i$  and the denominator counts the maximum number of links which could exist among them.

- **Diameter**

Evaluation of (a) the distribution of the shortest path (or geodesic) lengths  $d_{ij}$  of the graph, where  $d_{ij}$  is the length of the geodesic from node  $i$  to  $j$ ; (b) the diameter  $Diam(G)$ , which is the maximum value of  $d_{ij}$ ; (c) the matrix  $d$  of the shortest path lengths.

- **Betweenness node**

Evaluation of the Betweenness Centrality  $b_i$  of node  $i$ , also known as load, which is expressed as:

$$b_i = \frac{1}{(N - 1)(N - 2)} \sum_{j,k \in N, j \neq k} \frac{n_{jk}(i)}{n_{jk}} \quad (4)$$

where  $n_{jk}$  is the number of shortest paths connecting nodes  $j$  and  $k$  and  $n_{jk}(i)$  is the number of shortest paths between  $j$  and  $k$  passing through node  $i$ .

- **Betweenness edge**

Evaluation of the Betweenness Centrality  $e_{il}$  of the edge between nodes  $i$  and  $l$ , that is determined similarly to the node betweenness, except that  $n_{jk}(i)$  is substituted by the number of shortest paths between nodes  $j$  and  $k$  that pass through that edge.

- **Info Centr**

Evaluation of the Information Centrality  $I_i$  of node  $i$  which defines the importance of node  $i$  by the ability of the network to remedy the deactivation of that node. The network performance, before and after deactivation of a certain node, is evaluated by the efficiency  $E$  of the graph  $G$ :

$$E[G] = \frac{1}{N(N-1)} \sum_{i,j \in N, i \neq j} \frac{1}{d_{ij}} \quad (5)$$

that measures the mean flow-rate of information over  $G$ , and where  $d_{ij}$  is the geodesic from node  $i$  to  $j$ . The quantity  $E[G]$  varies in the range  $[0, 1]$ , and is finite also for disconnected graphs. The information centrality of node  $i$  is defined as the relative drop in the network efficiency caused by the removal from  $G$  of the edges incident to  $i$ :

$$I_i = \frac{E[G] - E[G_i]}{E[G]} \quad (6)$$

where  $G_i$  is the graph with  $N$  nodes and  $K - k_i$  edges obtained by removing from the original graph  $G$  the edges incident to node  $i$ .

- **Kshell**

Evaluation of the k-shell decomposition of the graph. Given a graph  $G = G(N, K)$  with  $N$  nodes and  $K$  edges, a  $k$ -core (or core of order  $k$ ) is defined through the subgraph  $H(C, K(C))$  induced by the set  $C \in N$  such that for each node  $n$  of  $C$  the  $degree(n) \leq k$ , and  $H$  is the maximum subgraph with this property. The k-shell decomposition is the recursively removal of all the nodes of degree  $k$  (or fewer) until all nodes of the remaining subgraph have at least degree  $k + 1$ , i.e. the subgraph is a  $(k + 1)$ -core. The decomposition begins with  $k$  equal to the minimum degree of the starting graph and carries on increasing  $k$  at each step. The process stops when no further nodes remain; the last not empty  $k$ -core provides a very robust definition of the heart (or nucleus) of a graph.

- **Community Structure**

The Girvan-Newman algorithm is one of the methods used to detect communities in complex systems. The notion of a "community structure" is related to that of clustering, though it isn't quite the same. A *community* consists of a subset of nodes within which the node-node connections are dense, and the edges to nodes in other communities are less dense. There are numerous alternative methods for detecting

communities in networks. These include hierarchical clustering, partitioning graphs to maximize quality functions such as graph modularity, k-clique percolation, etc.

The hierarchical clustering method is based. The edges with the greatest weights within the community are the most central ones. Although traditional in community detection, the method presents some pathologies. One of them for instance, is the inability to classify in a community a node which is connected to the network with only one edge. The Girvan-Newman algorithm, differently from other methods based on the assignement of a weight for every edge and placing these edges into an initially empty network (starting from edges with strong weights and progressing towards the weakest ones), works the opposite way. Instead of trying to construct a measure that tells us which edges are the most central to communities, it focuses on these edges that are least central, the edges that are most "between" communities. The communities are detected by progressively removing edges from the original graph, rather than by adding the strongest edges to an initially empty network.

Vertex betweenness has been studied in the past as a measure of the centrality and influence of nodes in networks. For any node  $i$ , *vertex betweenness* is defined as the number of shortest paths between pairs of nodes that run through it. It is a measure of the influence of a node over the flow of information between other nodes, especially in cases where information flow over a network primarily follows the shortest available path. The Girvan-Newman algorithm extends this definition to the case of edges, defining the "edge betweenness" of an edge as the number of shortest paths between pairs of nodes that run along it. If there is more than one shortest path between a pair of nodes, each path is assigned equal weight such that the total weight of all of the paths is equal to unity. If a network contains communities or groups that are only loosely connected by a few intergroup edges, then all shortest paths between different communities must go along one of these few edges. Thus, the edges connecting communities will have high edge betweenness (at least one of them). By removing these edges, the groups are separated from one another and so the underlying community structure of the network is revealed. The algorithm's steps for community detection are summarized below

- The betweenness of all existing edges in the network is calculated

- first.
- The edges with the highest betweenness are removed.
  - The betweenness of all edges affected by the removal is recalculated.
  - Steps 2 and 3 are repeated until no edges remain.

The fact that the only betweennesses being recalculated are only the ones which are affected by the removal, may lessen the running time of the process' simulation in computers. However, the betweenness centrality must be recalculated with each step, or severe errors occur. The reason is that the network adapts itself to the new conditions set after the edge removal. For instance, if two communities are connected by more than one edge, then there is no guarantee that all of these edges will have high betweenness. According to the method, we know that at least one of them will have, but nothing more than that is known. By recalculating betweennesses after the removal of each edge, it is ensured that at least one of the remaining edges between two communities will always have a high value. The end result of the Girvan-Newman algorithm is a dendrogram. As the Girvan-Newman algorithm runs, the dendrogram is produced from the top down (ie. the network splits up into different communities with the successive removal of links). The leaves of the dendrogram are individual nodes.

## 2 Usage

NAT can be used via registration and authentication at the web site *irriis.nat.ylichron.it*; this page could be also accessed through the project IRRIS homepage *www.irriis.org*.

### 2.1 Input data

- The query network must be undirected, resulting in a symmetric Adjacency matrix such that for its elements:  $a_{ij} = a_{ji}$ .
- It is required to number the network nodes with ordinal numbers starting from zero. It is also granted to start from 1, but it will cause the renumbering of the nodes with a -1 shifting.
- A text file (.ndf) with each line reporting a single couple of numbers, which identify linked nodes, is accepted. Each pair of numbers can be separated by space(s) and/or tab(s).

- Before the list of linked nodes, comment lines can be included, but each one must begin with character ">" or "#".
- Blank lines will be ignored, thus they can be present without restrictions.

## 2.2 Output data

Analysis replies can be as text (.txt) and/or figure (.jpg) and all together are zipped in a file (.zip) sent by e-mail as attachment. The adopted program zip (version 2.3) is a compression and file packaging utility for Unix, VMS, MS-DOS, OS/2, Windows NT, Minix, Atari and Macintosh, Amiga and Acorn RISC OS. It is analogous to a combination of the UNIX commands "tar" and "compress" and is compatible with PKZIP (Phil Katz's ZIP) for MSDOS systems. The extraction of files from the sent .zip archive can be performed by the most common compression/decompression programs such as unzip, pkunzip, winzip, winrar, etc.

The output files generated by the different analysis tools are the following:

### Network Image

Image (only for networks with nodes  $\leq 500$ ) comes as jpg file named aiSee.jpg. The aiSee.jpg file shows the picture of the submitted network where the nodes are colored in blue and the thickness of the edges is proportional to their weight.

### Degree Distribution

Results come as text and jpg files named, respectively, Degree\_Distr.txt and Degree\_Distr.jpg. For a network with a maximum degree  $K$ , the Degree\_Distr.txt file is composed of  $K + 1$  lines, each containing two space-separated numbers, which refer to, respectively, the degree number (from zero to  $K$ ) and the frequency (or probability) of that degree in the submitted network. The Degree\_Distr.jpg file shows the graph of the degree distribution in a log/log scale.

### Min-Cut

Results come as text (only for networks with nodes  $\leq 10000$ ) and jpg (only for networks with nodes  $\leq 500$ ) files named, respectively, SubNets.txt and aiSee.jpg. The SubNets.txt file is partitioned in 4 sections reporting: 1)

for each node, the components of eigenvectors (EigenVector1 and EigenVector2) corresponding to the two smallest eigenvalues; 2) the node numbers of the subnet  $A$  (with positive eigenvalues); 3) the node numbers of the subnet  $B$  (with negative eigenvalues); 4) the list of the node pairs connecting the two subnets. The aiSee.jpg file shows the picture of the submitted network, where the subnets  $A$  and  $B$  are colored respectively in red and blue, and the edges connecting the two subnets are evidenced through bold lines.

### **Graph Spectrum**

Results (only for networks with nodes  $\leq 10000$ ) come as a jpg file named Graph\_Spectr.jpg. The Graph\_Spectr.jpg file shows the graph of the occurrences of the calculated eigenvalues as a function of the same eigenvalues.

### **Shortest Paths Distribution**

Results come as text and jpg files named, respectively, Short\_Path\_Distr.txt and Short\_Path\_Distr.jpg. For a network with a max shortest path  $D$  (called Diameter), the Short\_Path\_Distr.txt file, after a section reporting the Diameter value, is composed of  $D + 1$  lines, each containing two space-separated numbers, which refer to, in order, the Shortest Path length (from 0 to  $D$ ) and the occurrences of that Shortest Path length in the query network. The Short\_Path\_Distr.jpg file shows the graph of the Shortest Paths Distribution in which the value  $\log N / \log(\langle k \rangle)$  (where  $N$  and  $\langle k \rangle$  are, respectively, the total number of nodes and the mean degree of the network) is evidenced with a vertical line.

### **Shortest Path Lengths Matrix**

Results come as text file named Short\_Path\_Matrix.txt. For a network of  $N$  nodes, the Short\_Path\_Matrix.txt file contains an  $N \times N$  matrix reporting all the shortest path lengths of the submitted network.

### **Clustering Coefficient**

Results come as text and jpg files named, respectively, Clustering\_Co.txt and Clustering\_Co.jpg. For a network of  $N$  nodes, the Clustering\_Co.txt file is composed, after a section reporting the Clustering Coefficient  $C$  of the whole network, of  $N$  lines, each containing two space-separated numbers,

which refer to, in order, the node number and the relative Clustering Coefficient  $c$  (varying in the range  $[0,1]$ ). The Clustering\_Co.jpg file shows the plot of the Node Clustering Coefficient versus node number.

### **Node Betweenness Centrality**

Results (only for networks with nodes  $\leq 10000$ ) come as text and jpg files named, respectively, Betweenness\_Node.txt and Betweenness\_Node.jpg. For a network of  $N$  nodes, the Betweenness\_Node.txt file is composed of  $N$  lines, each containing two space-separated numbers, which correspond to, in order, the node number and the relative Betweenness Centrality (varying in the range  $[0,1]$ ). The Betweenness\_Node.jpg file shows the plot of the Node Betweenness Centrality versus node number.

### **Edge Betweenness Centrality**

Results (only for networks with nodes  $\leq 1000$ ) come as a text file named Betweenness\_Edge.txt. For a network of  $K$  edges, the Betweenness\_Edge.txt file is composed of  $K$  lines, each containing three space-separated numbers: the first two are the numbers of the edge nodes and the third is the corresponding Edge Betweenness Centrality (varying in the range  $[0,1]$ ).

### **Information Centrality**

Results (only for networks with nodes  $\leq 1000$ ) come as text and jpg files named, respectively, Information\_Centr.txt and Information\_Centr.jpg. For a network of  $N$  nodes, the Information\_Centr.txt is composed of  $N$  lines, each containing two space-separated numbers, which correspond to, in order, the node number and the relative Information Centrality (varying in the range  $[0,1]$ ). The Information\_Centr.jpg file shows the plot of the Information Centrality versus node number.

### **K-Pruning**

Results come as text and jpg files named, respectively, Kshell\_Resume.txt and Kshell.jpg. The Kshell\_Resume.txt file reports the total number of nodes and the node identifier numbers of each  $(k+1)$ -core network resulting after each  $k$ -shell decomposition, starting from  $k$  equal to the minimum degree of the query network and ending with the  $k$  leading to a  $(k+1)$ -core

whose further pruning would delete all its nodes. Here is reported an example:

+++++++K-CORE = 2 +++++++

Matrix reduced to 6 Nodes:

Remaining Nodes:

4  
5  
6  
7  
8  
12

The Kshell.jpg file shows the plot of the number of nodes deleted during the  $k$  - shell decomposition as a function of the degree  $k$ .

## Part IV

# Topology Analysis of Weighted Networks

## 1 Topological properties of a weighted graph

The tool for the topological analysis of weighted networks of the **NAT** suite enables the user to evaluate different topological properties of a user-defined weighted graph which can be uploaded on the server web site from the user repository. The topological properties evaluated by such specific **NAT** tool are the following:

- **Connectdeness**

Verifies if the graph is connected or disconnected. A graph is defined as connected if there is a path connecting every pair of nodes  $i$  and  $j$  (with  $i \neq j$ ). Moreover, in case of a disconnected graph, the tool assesses the subgraphs whose nodes are completely connected.

- **Degree Distribution**

Calculates:

- (a) the degree distribution  $P(k)$ , which is the fraction of nodes in the graph displaying degree  $k$ . The degree  $k_i$  of a node  $i$  is the number of edges incident to that node.
- (b) the node weighted connectivity (or strength). The strength  $s_i$  of a node  $i$  is defined as:

$$s_i = \sum_{j \in N} w_{ij} \quad (1)$$

where  $N$  is the number of nodes of the graph and  $w_{ij}$  identifies the values (weights) associated to the links.

- **Clustering**

Calculates the weighted clustering coefficient  $C^W$ , deriving from the average, over all the nodes of the graph, of the local weighted clustering coefficient  $c_i^W$ :

$$C^W = \langle c^W \rangle = \frac{1}{N} \sum_{i \in N} c_i^W \quad (2)$$

The value of  $c_i^W$  is calculated, according to Barrat *et al.* (2004), as:

$$c_i^W = \frac{1}{s_i(n_i - 1)} \sum_{j,m \in n_i} \frac{(w_{ij} + w_{im})}{2} a_{ij} a_{jm} a_{mi} \quad (3)$$

where  $s_i(n_i - 1)$  is a normalization factor, ensuring that  $0 \leq C_i^W \leq 1$ , in which  $s_i$  is the weighted connectivity of node  $i$  and  $n_i$  is the number of nodes adjacent to  $i$ ;  $w_{ij}$  and  $w_{im}$  are the weights associated to the links between nodes  $i$  and  $j$  and between nodes  $i$  and  $m$ , respectively.

- **Diameter**

Calculates:

(a) the weighted shortest path lengths  $d_{ij}$  of the graph, where  $d_{ij}$  is the smallest sum of the edge lengths throughout all the possible paths in the graph from node  $i$  to  $j$ .

(b) the diameter, which is the maximum  $d_{ij}$ , and the average of the weighted shortest paths.

(c) the node closeness centrality  $C^C$ , which quantifies the closeness of a node  $i$  to all the other nodes, defined as:

$$C_i^C = \frac{N - 1}{\sum_{i \in N, i \neq j} d_{ij}} \quad (4)$$

- **Weighted Betweenness Centrality**

Calculates the weighted betweenness of nodes and edges. The weighted betweenness  $b_i$  of a node  $i$  is expressed as:

$$b_i = \frac{n_i}{(N - 1)(N - 2)} \quad (5)$$

where  $n_i$  is the number of weighted shortest paths containing node  $i$ . The weighted betweenness of an edge is determined similarly to the node betweenness, except that  $n_i$  is substituted by the number of weighted shortest paths  $n_{ij}$  that pass through that edge.

- **Node Information Centrality**

Calculates the information centrality of the nodes. Information centrality  $C^I$  assesses a node importance by the ability of the network to remedy the deactivation of that node. The network performance, before and after deactivation of a certain node, is evaluated by the

efficiency  $E$  of the graph  $G$ :

$$E[G] = \frac{1}{N(N-1)} \sum_{i,j \in N, i \neq j} \frac{1}{d_{ij}} \quad (6)$$

that measures the mean flow-rate of information over  $G$ , and where  $d_{ij}$  is the weighted shortest path from node  $i$  to  $j$ . The quantity  $E[G]$  varies in the range  $[0,1]$ , and is finite also for disconnected graphs. The information centrality  $C^I$  of node  $i$  is defined as the relative drop in the network efficiency caused by the removal from  $G$  of the edges incident to  $i$ :

$$C_i^I = \frac{E[G] - E[G'_i]}{E[G]} \quad (7)$$

where  $G'_i$  is the graph with  $N$  nodes and  $K - k_i$  edges obtained by removing from the original graph  $G$  the edges  $k_i$  incident to node  $i$ .

- **Edge Information Centrality**

Calculates the information centrality of the edges. Information centrality  $C^I$  assesses an edge importance by the ability of the network to remedy the deactivation of that edge. The network performance, before and after deactivation of a certain edge, is evaluated, as for a node deactivation, by the efficiency  $E$  of the graph  $G$ . The information centrality  $C^I$  of edge  $k_{ij}$  is defined as the relative drop in the network efficiency caused by the removal from  $G$  of the edge between nodes  $i$  and  $j$ :

$$C_{ij}^I = \frac{E[G] - E[G'_{ij}]}{E[G]} \quad (8)$$

where  $G'_{ij}$  is the graph with  $N$  nodes and  $K - 1$  edges obtained by removing from the original graph  $G$  the edge between nodes  $i$  and  $j$ .

- **Weighted Min-Cut**

Evaluation of the min-cut, consisting in the optimal partitioning of a graph of  $N$  nodes into two connected subgraphs of dimensions  $N_1$  and  $N_2$  (such that  $N = N_1 + N_2$ ) separated by a minimum number of  $K$  edges. In the partitioning the weight of the edge is also considered.

## 2 Usage

NAT can be used via registration and authentication at the web site [irriis.nat.ylichron.it](http://irriis.nat.ylichron.it); this page could be also accessed through the project IRRIS homepage [www.irriis.org](http://www.irriis.org).

## 2.1 Input data

- It is required to number the network nodes with ordinal numbers starting from zero or 1 without continuity solution.
- A text file (.ndf) with each line reporting a single triple of numbers, the first two identifying the linked nodes and the third corresponding to the weight of the link, is accepted. The numbers of each triple must be separated by space(s) and/or tab(s).
- Before the list of linked nodes and weights, comment lines can be included, but each one must begin with character ">" or "#".
- Blank lines will be ignored, thus their inclusion is unrestricted.

## 2.2 Output data

Analysis replies can be as text (.txt) and/or figure (.jpg) and all together are zipped in a file (.zip) sent by e-mail as attachment. The adopted program zip (version 2.3) is a compression and file packaging utility for Unix, VMS, MS-DOS, OS/2, Windows NT, Minix, Atari and Macintosh, Amiga and Acorn RISC OS. It is analogous to a combination of the UNIX commands "tar" and "compress" and is compatible with PKZIP (Phil Katz's ZIP) for MSDOS systems. The extraction of files from the sent .zip archive can be performed by the most common compression/decompression programs such as unzip, pkunzip, winzip, winrar, etc.

The output files generated by the different analysis tools are the following:

### Network Image

Image (only for networks with nodes  $\leq 500$ ) comes as jpg file named aiSee.jpg. The aiSee.jpg file shows the picture of the submitted network where the nodes are colored in blue and the thickness of the edges is proportional to their weight.

### Connectdeness

Results come as text file named Sub\_Graphs.txt. For a disconnected network  $G$ , the file reports the subnetworks  $\subset G$  whose nodes are connected, listing the node numbers. Otherwise, if the network is connected, the Sub\_Graphs.txt reports the list of all the nodes of the submitted network.

## Degree Distribution

Results come as text and jpg files named, respectively, Degree\_Distr.txt, Weigh\_Connect.txt and Degree\_Distr.jpg. For a network with a maximum degree  $K$ , the Degree\_Distr.txt file is composed of  $K + 1$  lines, each containing two space-separated numbers, which refer to, respectively, the degree number (from zero to  $K$ ) and the frequency (or probability) of that degree in the submitted network. For a network of  $N$  nodes, the Weigh\_Connect.txt file is composed of  $N$  lines, each containing three space-separated numbers, which correspond to, in order, the node number, the node degree, and the relative weighted connectivity (or strength). The Degree\_Distr.jpg file shows the graph of the degree distribution.

## Weighted Clustering Coefficient

Results come as text and jpg files named, respectively, Weigh\_Clust\_Co.txt and Weigh\_Clust\_Co.jpg. For a network of  $N$  nodes, the Weigh\_Clust\_Co.txt file, after a section reporting the Weighted Clustering Coefficient  $C$  of the whole network, is composed of  $N$  lines, each containing two space-separated numbers, which refer to, in order, the node number and the relative Weighted Clustering Coefficient  $c$  (varying in the range  $[0,1]$ ). The Weigh\_Clust\_Co.jpg file shows the bar plot of the Weighted Node Clustering Coefficient distribution.

## Closeness Centrality

Results come as text file named Closeness\_Cen.txt. For a network of  $N$  nodes, the Closeness\_Cen.txt file, after a section reporting the Diameter and the average of the weighted shortest paths, is composed of  $N$  lines, each containing two space-separated numbers, which refer to, in order, the node number and the relative Closeness Centrality.

## Shortest Path Lengths

Results (only for networks with nodes  $\leq 500$ ) come as text file named Paths.txt. For every pair of nodes  $i$  and  $j$  (with  $i \neq j$ ) of the submitted network, the Paths.txt file reports the value of the shortest path length and the node sequence of the path.

### **Weighted Node and Edge Betweenness Centrality**

Results (only for networks with nodes  $\leq 10000$ ) come as text and jpg files named, respectively, `Weigh_Node_Betw.txt`, `Weigh_Edge_Betw.txt` and `Weigh_Node_Betw.jpg`. For a network of  $N$  nodes, the `Weigh_Node_Betw.txt` file is composed of  $N$  lines, each containing two space-separated numbers, which correspond to, in order, the node number and the relative Weighted Node Betweenness Centrality (varying in the range  $[0,1]$ ). For a network of  $K$  edges, the `Weigh_Edge_Betw.txt` file is composed of  $K$  lines, each containing three space-separated numbers: the first two are the numbers of the edge nodes and the third is the corresponding Weighted Edge Betweenness Centrality (varying in the range  $[0,1]$ ). The `Weigh_Node_Betw.jpg` file shows the bar plot of the Weighted Node Betweenness Centrality distribution.

### **Weighted Node Information Centrality**

Results (only for networks with nodes  $\leq 1000$ ) come as text and jpg files named, respectively, `Weigh_Node_Inf_Cen.txt` and `Weigh_Node_Inf_Cen.jpg`. For a network of  $N$  nodes, the `Weigh_Node_Inf_Cen.txt` file is composed of  $N$  lines, each containing two space-separated numbers, which correspond to, in order, the node number and the relative Weighted Information Centrality (varying in the range  $[0,1]$ ). The `Weigh_Node_Inf_Cen.jpg` file shows the bar plot of the Weighted Node Information Centrality distribution.

### **Weighted Edge Information Centrality**

Results (only for networks with nodes  $\leq 1000$ ) come as text file named `Weigh_Edge_Inf_Cen.txt`. For a network of  $K$  edges, the `Weigh_Edge_Inf_Cen.txt` file is composed of  $K$  lines, each containing three space-separated numbers: the first two are the numbers of the edge nodes and the third is the corresponding Weighted Edge Information Centrality (varying in the range  $[0,1]$ ).

### **Weighted Min-Cut**

Results come as text (only for networks with nodes  $\leq 10000$ ) and jpg (only for networks with nodes  $\leq 500$ ) files named, respectively, `MinCut.txt` and `aiSee-MC.jpg`. The `MinCut.txt` file is partitioned in 4 sections reporting: 1) for each node, the components of eigenvectors (`EigenVector1` and `EigenVector2`) corresponding to the two smallest eigenvalues; 2) the node numbers

of the subnet  $A$  (with positive eigenvalues); 3) the node numbers of the subnet  $B$  (with negative eigenvalues); 4) the list of the node pairs connecting the two subnets. The aiSee-MC.jpg file shows the picture of the submitted network, where the subnets  $A$  and  $B$  are colored respectively in red and blue, and the edges connecting the two subnets are evidenced through bold lines.

## Part V

# Communication Networks Traffic Simulator

The **NAT** Communication Networks Traffic Simulator tool enables the user to simulate the traffic dynamic in a packet switched communication network. Packet switched networks have a crucial relevance in our world; they are the backbone for all our communications and a lots of critical infrastructure relies on them. The aim of this tool is to permit the study of the influence of network's structure in the traffic dynamics; therefore the **NAT** Communication Networks Traffic Simulator tool permits to reproduce the traffic dynamics of a "real" communication infrastructure. The simulator is able to model all physical elements of a packet switched network: routers, links and packets, and reproduce some networks' basic features such as: the capability to send a data packet from an origin to a given destination, a routing table system and a FIFO routing strategy. To permit the study of the influence of topology on the network traffic management capabilities some of traffic control mechanism such as TCP/IP's flow control algorithms were be omitted.

## 1 Traffic Simulation Models

The simulator takes as input the network topology as an undirected unweighted graph, in which, each node represents a router of the network and, each edge represents a connection between two routers (link), along which data can be exchanged in data unit referenced to as *packet*. Routers are identified by an *identification number* (ID)  $i$ , ( $0 \leq i \leq (N - 1)$ ), where  $N$  is the total number of routers in the network.

The dynamical evolution in time is discretized in *Time Step(s)* (TS). At each TS, a router can send a packet to another router under a certain probability. A node cannot send a packet to itself. The amount of traffic present in the network is measured by the variable  $\lambda$  ( $0 \leq \lambda \leq 1$ ) which measures the frequency with which a node emits a packet (or, equivalently, the fraction of nodes that, at a given TS, emits a packet). The packet is generated by a randomly chosen "emitting" node and directed to a randomly chosen "destination" node. With this definition, for instance,  $\lambda = 0.1$  represents a level of traffic where, at each time step, 10% of the  $N$  nodes of the network generates a packet of data directed toward an equal number of destination

nodes. The two sets of nodes could be similar or different; the only forbidden action is that of a node sending a packet to itself.

Each router hosts a routing table (RT) where information about the next hop to reach each other router in the network is stored. The RTs present on each node are evaluated, once forever, via the *Breadth-first search* algorithm. The strategy which rules packet dispatching is based, in fact, on the shortest path between sender and receiver: each node's pair (origin-destination) is associated to a routing path which is formed by the nodes joining the minimal-distance path between the two nodes.

In the present model, at a given TS, a node can send a fixed number of data packets (see sec.1.2) and, in turn, can receive as many packets as needed (i.e. if more than one neighbor sent it a packet at the previous TS). In order to host multiple packets arriving simultaneously and to dispatching them at later times, each node has an infinite-size buffer, where packets are stored. When a packet arrives to a router, two cases are taken into account:

- if the router is the packet's destination, the packet is eliminated from the network - it is supposed to be forwarded by an *intra-AS* routing to the destination host within the AS represented by that router.
- otherwise, the packet is stored in the last position of the router's buffer where it waits to be delivered.

The buffer management complies with the *First In - First Out* (FIFO) policy: at each TS, each router picks the first packet (in order of arrival) from its queue and forwards it according to its routing table. All the remaining packets get ahead in the queue. Several aspects need to be pointed out:

- two or more packets can reach the router at the same TS; in case one or more have not reached the destination, they are all stored into the buffer and they will be ordered according with the provenance router (lowest ID, higher yield).
- each link between two routers is traversed in a TS: hence, every packet hops from a router to the next in the same TS.
- a packet cannot traverse more than one link in a TS.
- a router can forward a fixed number of packets from its queue in a TS.

Once a packet is emitted, it is immediately put into its emitter's buffer; in case this buffer is empty, the packet is forwarded during the same TS it has been generated.

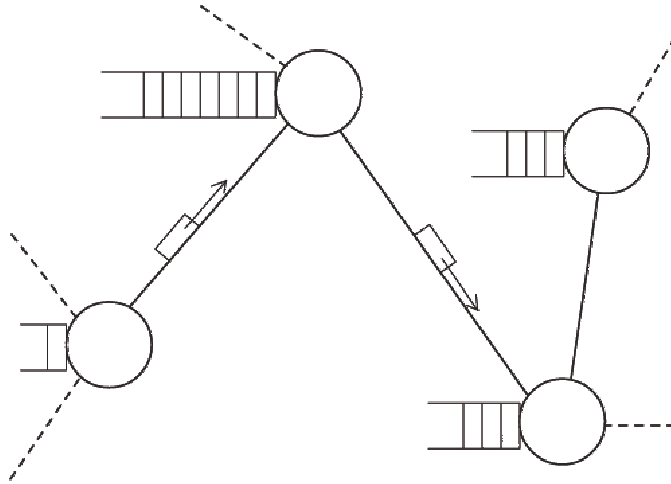


Figure 4: *visualization of routers with their respective buffers and packet being sent.*

Simulation starts with an “empty” network, i.e. with no traffic and all buffers empty. As soon as simulation progresses, buffers start filling and emitted packets take a certain time (i.e. a certain number of simulation TS) to reach the destination node. In the best possible case, the packet is re-emitted by a node the TS after its arrival to that node. However, if the buffer of that node is already filled by packets arrived at earlier times, the packet must remain, on that node, a number of TS equal to the number of packets waiting in the buffer, according to the FIFO policy.

### 1.1 Routers’ Model

In “real world systems” routers have different performances according to their role in the network: an highly connected router, which have to manage high quantity of traffic, most probably will be more powerful than a poorly connected one. To reproduce such situation in the simulator, the **Power Routers Models** are now introduced. In the **Power Router Models** the number of sending operations that a router  $i$  could perform at each TS is function of his degree. To control the number of **Power Routers** in the network a threshold is introduced, and also the threshold is function of the degree of a node; if the router degree is greater than the threshold the router is considered a power router and it could perform more sending operations per TS, else if the router degree is less than the threshold the router is

considered a normal router and it could perform only one sending per TS. The threshold is computed as follow:

$$k_i > \langle k \rangle b \quad (1)$$

Where  $k_i$  is the degree of the generic router  $i$ ,  $\langle k \rangle$  is the average degree of routers in the network and  $b$  is an adjustable parameter to control the number of **Power Routers**. The number of sending operations permitted from each router is computed as follow:

$$\theta_i = \frac{k_i}{\langle k \rangle} a \quad (2)$$

Where  $\theta_i$  is the number of sending operations and  $a$  is an adjustable parameter of the system.

## 1.2 Routing

When a packet must be forwarded, the next router is selected so that the packet is delivered to its destination along the shortest path. If more than one candidate to the next hop exists, a strategy is needed to select the recipient. In our simulations we consider three routing strategies: the **fixed routing**, the **deterministic routing** and the **probabilistic routing**.

The difference is based on the number of paths that routers take into account and on the way they choose on which one they will forward a packet.

In the **fixed routing**, each router has only one possible neighbor node associated to a given destination node. In other words, the RT of node  $i$  associates, to each destination node  $j$ , one and only one node, say  $k$  belonging its neighbors.

In the other two strategies, more than a single shortest path can be selected. In case more than one shortest path exist for a couple of nodes  $(i, j)$ , the routing table of  $i$  will have two choices (i.e. two links) associated to the node  $j$ . Note that more than two choices could exist: we found that for an artificially generated 3000 nodes Scale Free network - Internet-Like) the average number of possible choices for each couple of nodes is 1.58721, while for a Random network of the same size is 1.89529.

How does the router choose the node where the packet must be forwarded?

In order to introduce the **deterministic routing** and the **probabilistic routing**, we must introduce a “routing probability function”. When a router has to forward a packet choosing between two routes  $A$  and  $B$  based on the

destination address, we assign the probability to choose  $A$  and  $B$  by the following equation:

$$P(A) = \frac{e^{-\beta X_A}}{e^{-\beta X_A} + e^{-\beta X_B}}, \quad (3)$$

$$P(B) = \frac{e^{-\beta X_B}}{e^{-\beta X_A} + e^{-\beta X_B}}, \quad (4)$$

$$P(A) + P(B) = 1, \quad (5)$$

where  $\beta$  is an adjustable parameter,  $X_A$  and  $X_B$  are the number of packets that have already traversed the routes  $A$  and  $B$ , representing thus the usage of the link. In our mechanism, we evaluate these probabilities by evaluating firstly  $P(A)$  and then  $P(B) = 1 - P(A)$ . We can use the parameter  $\beta$  to drive the probabilities.

- if  $\beta \rightarrow 0$ , both of the exponential terms in the equations 3 and 4 tends to 1 and both  $P(A)$  and  $P(B)$  tends to 0.5. In such a case the route is randomly chosen with no relevance accorded to the usage of the two links. This is what we call **probabilistic routing**.
- If  $\beta \rightarrow 1$ , the usage of the routes is more important in the evaluation of the two probabilities, making bigger the probability of the less used link. This is what we call **deterministic routing**.
- If  $\beta \rightarrow \infty$ ,  $P(A)$  always tends to 0, thus recalling the **fixed routing** by always choosing the route  $B$ .

The variation of  $\beta$  between 0 and 1 determines the degree of randomness of the routing, and even if  $X_A > X_B$  there's a probability of routing a packet through  $B$ ; this probability decreases as the difference of usage between  $A$  and  $B$  increases. Figure 5 shows an example of routing where a decision must be taken.

### 1.3 Traffic properties evaluated during the simulations

Different kind of technological observables are measured on the network during the simulations. These are carried out for a large number of TS (hereafter referred to as  $\tau$ ). The simulation time in TSs  $\tau$  is chosen to be enough large to allow the "exploration" of all the network, by all the generated packets, to allow a complete sampling of the network's topology.

The two main quantities which have been evaluated during the simulation time are:

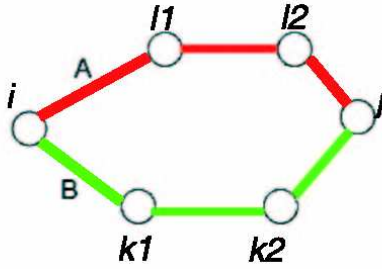


Figure 5: *example of possibility of different routing: two different shortest paths exist between nodes  $i$  and  $j$ .*

- the fraction  $p$  of delivered packets during the duration of the simulation;
- the average delivery time  $\langle T \rangle$  (packet *delivery time*) which is the time distance between the TS when the packet is produced and the TS when the packet arrives to its destination;

The most important quantity, which somehow summarizes the quality of functioning of the network and its efficiency is the packet's average delivery time  $\langle T \rangle$ , which has been evaluated as a function of the traffic level  $\lambda$ . Each generated packet, at the end of its path, arrives to the destination node. The measure of the overall time spent by the packet to perform its path from the emitting node  $i$  to the destination node  $j$ , provides a significant estimate of the capability of the network to sustain the traffic level. After all, the performance of a network is measured in terms of the time a data packet spent to be delivered. If  $t_k$  is the time needed to the packet  $k$  to travel from its emitting node to the destination node, then the required value of  $\langle T \rangle$  is

$$\langle T \rangle = \frac{1}{M} \sum_{k=1}^M t_k \quad (6)$$

where  $M$  is the number of packets which have been *effectively* delivered within the simulation time  $\tau$ . This definition accounts for the fact that, as we assume a finite simulation time, at its completion, only a fraction of emitted packets will have been effectively *delivered* to the destination node; the remaining fraction will still be in travel toward the destination node. Therefore the average of eq.6 is evaluated only on packets which have been completed their route up to the destination node during  $\tau$ . The others will not be taken into account in the average process, still they affect the network's behavior draining time and resources to be forwarded.

## 2 Usage

NAT's **Communication Networks Traffic Simulator** tool can be used via registration and authentication at the web site [irriis.nat.ylichron.it](http://irriis.nat.ylichron.it); this page could be also accessed through the project IRRIS homepage [www.irriis.org](http://www.irriis.org).

### 2.1 Input Data

- A valid e-mail address for the results. Once the simulation ended the results' files will be sent by e-mail to the user.
- A valid *Network Description File*. A text file (.ndf) with each line reporting a single couple of numbers, which identify linked nodes, is accepted. Each pair of numbers can be separated by space(s) and/or tab(s). It is required to number the network nodes with ordinal numbers starting from zero or 1 without continuity solution. Blank lines will be ignored, thus their inclusion is unrestricted.
- The initial value of  $\lambda$  parameter. It defines the traffic level from which the simulation starts. The must be a decimal number included between 0 and 1 ( $0 < \lambda < 1$ ). Decimal digits must be separated from integer digits by a dot "." (example 0.001).
- The final value of  $\lambda$  parameter. It defines the traffic level where the simulation stops. The must be a decimal number included between 0 and 1 ( $0 < \lambda < 1$ ). Decimal digits must be separated from integer digits by a dot "." (example 0.001). This value must be greater than the initial value of  $\lambda$ .
- Simulation Model. The Communication Networks Traffic Simulator tool provides seven different simulation models. It's possible to choose between the three routing strategies:
  - *Fixed*  $\beta = \infty$
  - *Deterministic*  $\beta = 1$
  - *Probabilistic*  $\beta = 0$

and, for each routing strategy, two options for the power of in-homogeneous routers:

- *High power routers* with  $a = 2$  and  $b = 6$

– *Low power routers* with  $a = 1.5$  and  $b = 20$

It's also possible to select a “simplified” model with fixed routing and homogeneous routers (each router can forward only a packet per TS).

- Run name, a name to distinguish different Runs. It must be comprised between 3 and 8 characters and couldn't include some special symbols: *blank spaces*,  $\backslash$ ,  $/$ ,  $*$ ,  $?$ ,  $:$ ,  $"$ ,  $<$ ,  $>$ ,  $|$ .

## 2.2 Output Data

Simulation replies consist of a text (**results.txt**), a figure (**results.jpg**) and a log (simulation.log) file all together are zipped in a file (.zip) sent by e-mail as attachment. The adopted program zip (version 2.3) is a compression and file packaging utility for Unix, VMS, MS-DOS, OS/2, Windows NT, Minix, Atari and Macintosh, Amiga and Acorn RISC OS. It is analogous to a combination of the UNIX commands “tar” and “compress” and is compatible with PKZIP (Phil Katz's ZIP) for MSDOS systems. The extraction of files from the sent .zip archive can be performed by the most common compression/decompression programs such as unzip, pkunzip, winzip, winrar, etc.

### **results.txt**

The file consists of a set of lines, each line represents the results of a simulation for a fixed value of  $\lambda$ . Each line is composed of six columns organized as follows:

1.  $\lambda$ , the traffic level utilized in the simulation.
2.  $\langle T \rangle$ , the mean packets' arrival time.
3. Percentage of arrived packets, the number of arrived packets when the simulation ends in relation with the number of sended packets.
4. The value of the  $\beta$  parameter. Represents the routing strategy utilized
5. The value of the  $b$  parameter. Represents the threshold over which a router is considered a power router.
6. The value of the  $a$  parameter. Represents the number of sending operation that one power router could perform in a single TS.

$\lambda$	$\langle T \rangle$	% of arrived packets	$\beta$	$b$	$a$
0.001	5.68	100	1e+08	6	2
0.002	5.34545	100	1e+08	6	2
0.003	5.625	100	1e+08	6	2
0.004	5.51818	99.0991	1e+08	6	2
0.005	5.44526	99.2754	1e+08	6	2
0.006	5.38323	99.4048	1e+08	6	2
0.007	5.43005	98.9744	1e+08	6	2
0.008	5.38249	99.0868	1e+08	6	2
0.009	5.54878	99.1935	1e+08	6	2
0.01	5.51103	99.6337	1e+08	6	2

Table 2: *Example of output file structure.*

`results.jpg`

The file contains the graphical representation of results, in `.jpg` format. The  $x$  axis of the graph represents the value of  $\lambda$ , the  $y$  axis represents the mean packets arrival time  $\langle T \rangle$ , both axes are in logarithmic scale.

`simulation.log`

Contains simulations' logs information; in case of errors the file reports the cause of the error if known.

## Results

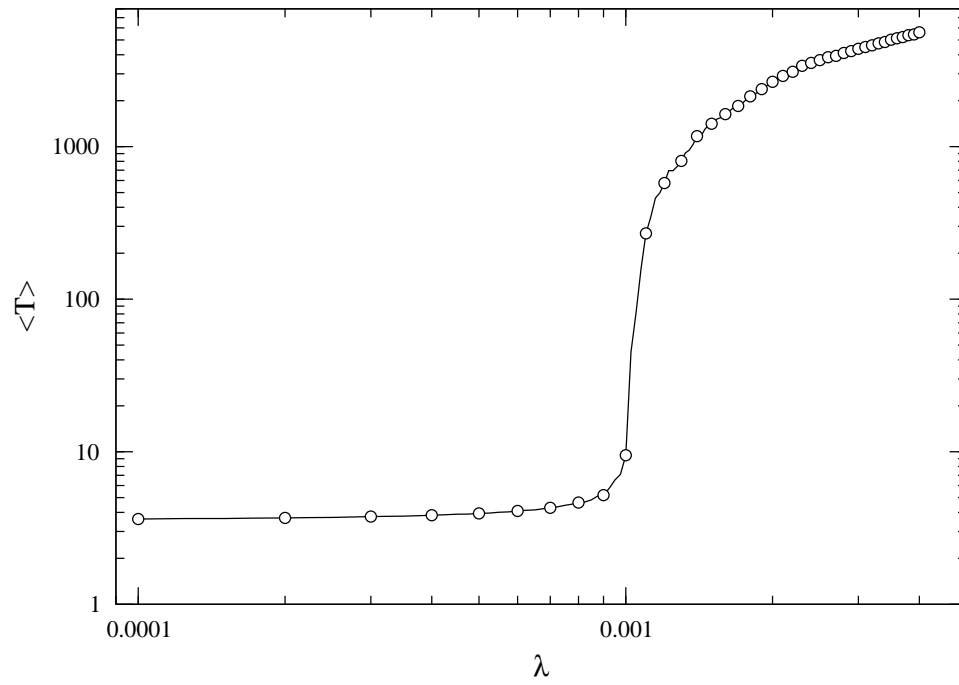


Figure 6: *example of graphical representation of results.*

## Part VI

# Vulnerability Analysis

## 1 Introduction

A major interest in the Critical Infrastructures analysis is related to the assessment of the *Vulnerability* of a network. The term *Vulnerability* indicates the proneness of the network to be damaged if one (or more) structural elements (such as nodes or links) are removed. In other terms, it is useful to evaluate the *Conditional Probability*  $P(n | n_k)$  than  $n$  nodes will result to be disconnected if  $n_k$  links are simultaneously removed. The value of the *Conditional Probability*  $P(n | n_k)$  does, indeed, depends on which link(s) is (are) removed. The tool will thus provide an average over a number of different choices of removed link(s).

The NAT tool allows, once the network is uploaded, to evaluate the *Conditional Probability*  $P(n | n_k)$ .

## 2 Usage

NAT can be used via registration and authentication at the web site [irriis.nat.ylichron.it](http://irriis.nat.ylichron.it); this page could be also accessed through the project IRRIS homepage [www.irriis.org](http://www.irriis.org).

### 2.1 Input data

- The query network must be undirected, resulting in a symmetric Adjacency matrix such that for its elements:  $a_{ij} = a_{ji}$ .
- It is required to number the network nodes with ordinal numbers starting from zero. It is also granted to start from 1, but it will cause the renumbering of the nodes with a -1 shifting.
- A text file (.ndf) with each line reporting a single couple of numbers, which identify linked nodes, is accepted. Each pair of numbers can be separated by space(s) and/or tab(s).
- Before the list of linked nodes, comment lines can be included, but each one must begin with character ">" or "#".

- Blank lines will be ignored, thus they can be present without restrictions.
- $n_k$ , the number of links which should be simultaneously removed (varying in the range [1,10]). When  $n_k = 1$  the tool will eliminate, one at a time, all network links and the result is a simple average over the  $k$  runs. When  $n_k > 1$ , the number of possible complexions of  $n_k = 2, 3, \dots$  would be exceedingly large to be exhaustively evaluated. Therefore, for a network of  $k$  links, the code will evaluate the probability as the average over the removal of a number of  $n_k$ -uples  $h$  as large as  $k \log_{10}(k)$ .

## 2.2 Output data

Analysis replies are in text (.txt) format zipped in a file (.zip) sent by e-mail as attachment. The adopted program zip (version 2.3) is a compression and file packaging utility for Unix, VMS, MS-DOS, OS/2, Windows NT, Minix, Atari and Macintosh, Amiga and Acorn RISC OS. It is analogous to a combination of the UNIX commands "tar" and "compress" and is compatible with PKZIP (Phil Katz's ZIP) for MSDOS systems. The extraction of the file from the sent .zip archive can be performed by the most common compression/decompression programs such as unzip, pkunzip, winzip, winrar, etc.

Results (only for networks with nodes  $\leq 1000$ ) come as a text file named Conditional\_Prob.txt. For a network of  $n$  nodes, the file is composed of  $L$  lines, each containing two space-separated numbers, which refer to, respectively, the number of disconnected nodes (varying in the range from zero to a maximum of  $n/2$ ) and the probability of that number after the removal of  $n_k$  links from the submitted network.

# Contents

<b>I</b>	<b>Introduction</b>	<b>1</b>
<b>II</b>	<b>Networks Builder</b>	<b>2</b>
<b>1</b>	<b>Types of Complex Networks</b>	<b>2</b>
<b>2</b>	<b>Usage</b>	<b>6</b>
2.1	Input Data . . . . .	6
2.2	Output Data . . . . .	7
<b>III</b>	<b>Topology Analysis of Unweighted Networks</b>	<b>9</b>
<b>1</b>	<b>Topological properties of an unweighted graph</b>	<b>9</b>
<b>2</b>	<b>Usage</b>	<b>13</b>
2.1	Input data . . . . .	13
2.2	Output data . . . . .	14
<b>IV</b>	<b>Topology Analysis of Weighted Networks</b>	<b>18</b>
<b>1</b>	<b>Topological properties of a weighted graph</b>	<b>18</b>
<b>2</b>	<b>Usage</b>	<b>20</b>
2.1	Input data . . . . .	21
2.2	Output data . . . . .	21
<b>V</b>	<b>Communication Networks Traffic Simulator</b>	<b>25</b>
<b>1</b>	<b>Traffic Simulation Models</b>	<b>25</b>
1.1	Routers' Model . . . . .	27
1.2	Routing . . . . .	28
1.3	Traffic properties evaluated during the simulations . . . . .	29

<b>2 Usage</b>	<b>31</b>
2.1 Input Data . . . . .	31
2.2 Output Data . . . . .	32
<b>VI Vulnerability Analysis</b>	<b>35</b>
<b>1 Introduction</b>	<b>35</b>
<b>2 Usage</b>	<b>35</b>
2.1 Input data . . . . .	35
2.2 Output data . . . . .	36