

Critical Infrastructure Protection in Homeland Security:

Defending a Networked Nation

Ted G. Lewis
tlewis@nps.edu

WHAT THIS MANUAL INCLUDES	1
CASCADENET	1
FTPLUS	5
INTERNETVIRUS	6
NETWORKANALYSIS	10
POWERGRIDSIM	13
RGB (TREELINK)	16
RNET	18
RSA	22
RTREE	24
SCALE-FREE(POWERGRAPH)	26
SHORTESTLINK	28
TERMITES	30

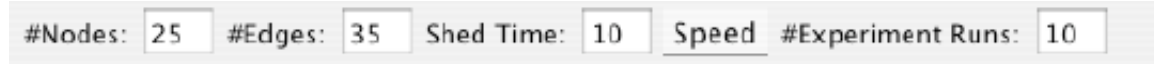
What This Manual Includes

Separate operations manuals exist for NetworkAnalysis and FTplus, because these elaborate programs form the core of this course. There are 10 other programs that do not have operation manuals. These programs are explained in this section.

CascadeNet

The web-based version of *CascadeNet* is called CascadeNet.html and the desktop (stand-alone) version is called CascadeNet.jar. It demonstrates the behavior of a cascade failure in an arbitrary network, and provides tools for studying the effects of network structure on its ability to resist cascade failure.

Input Fields



Input fields to CascadeNet are shown at the top of the display screen. #Nodes equals the number of nodes, #Edges equals the number of links, Shed Time is equal to the length of time it takes for a damaged node to be removed after it fails, and the Speed menu allows you to set the speed of the program. When you elect to run automatic experiments, #Experiment Runs is equal to the number of cascade failures that are averaged to get the each point in the graphs at the right.

Control Buttons



Control buttons are located along the bottom of the display screen. The SCALE-FREE button activates the emergence of a scale-free network using the organizing principle described in the book and used by program Scale-Free (PowerGraph). The PROTECT button allows you to select a node, press PROTECT-ON, and prevent the protected node from failing during a cascade failure. The SHED-ON button activates a failure by randomly selecting a node, marking it as failed (RED), and then removing it after Shed-Time time units (BLACK). Select SLOW in the SPEED menu to observe this.

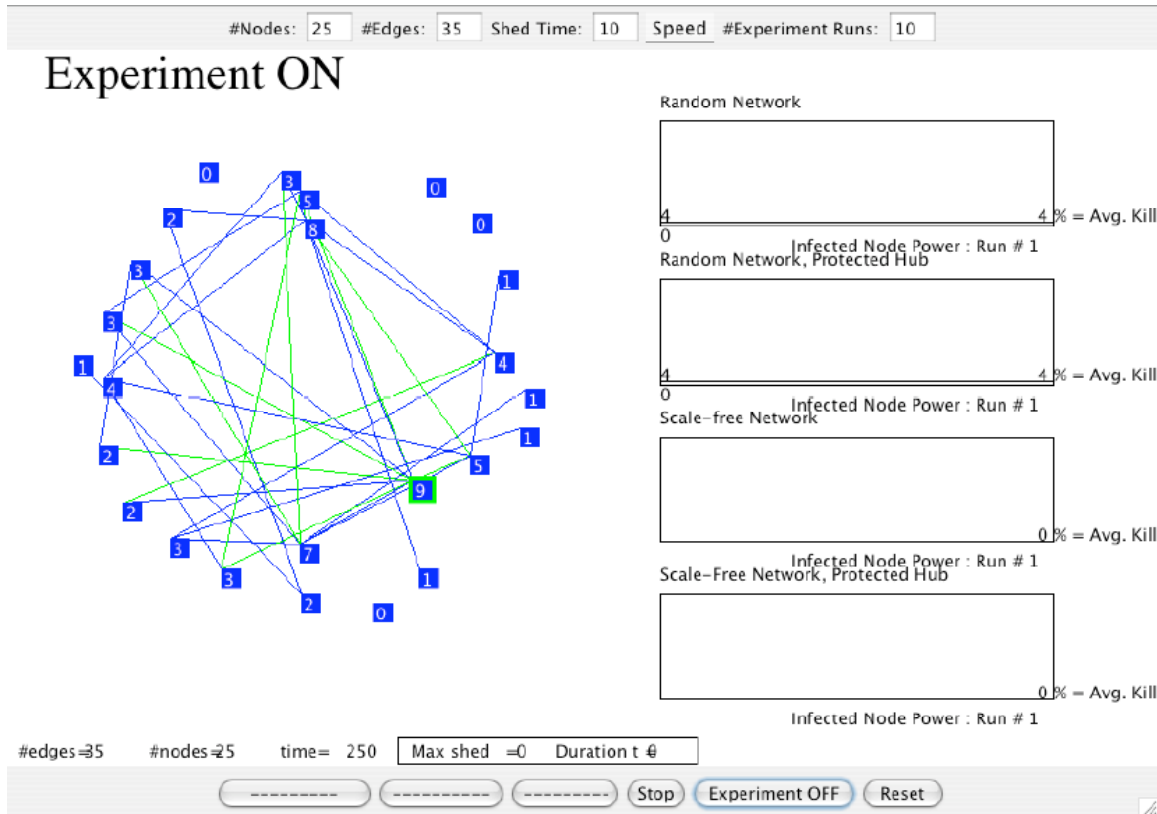
GO/STOP does the obvious. EXPERIMENT-ON disables the single cascade failure mode, and begins running 4 experiments, automatically. These experiments are explained, below. RESET stops the program and resets everything to the initial state, prior to pressing GO.

Here is how CascadeNet works: When the GO button is pressed, CascadeNet constructs a random network by randomly selecting pairs of nodes and linking them. It then waits for you to press SCALE-ON, SHED-ON or EXPERIMENT-ON. If SCALE-ON is pressed, CascadeNet emerges a scale-free network by preferential selection of nodes, using the increasing returns organizing principle described in the book.

When SHED-ON is pressed, CascadeNet selects a node at random and paints it RED, indicating that it is failing. CascadeNet spreads this failure to other nodes through adjacent links. Each failing node (RED) takes SHED TIME units to fail. After SHED TIME units have elapsed, the failing node is painted BLACK, indicating that it is being disconnected from the network. BLACK nodes are removed, along with their links. Therefore, the network begins to “come apart”, as the cascade failure spreads.

EXPERIMENTS

When EXPERIMENT-ON is pressed, CascadeNet changes modes. It stops whatever it was doing and begins to automatically run through four experiments, see the figure, below.



Note the three other control buttons are disabled, because they are meaningless while the experiments are automatically running. Second, the four graphical displays on the right side of the screen summarize the four experiments described in the book. Each experiment performs a random cascade network failure #EXPERIMENT RUNS times. So, if #EXPERIMENT RUNS equals 10, then each data point displayed in a graph corresponds to a cascade failure. The AVG line of each graph summarizes these values by averaging over them.

The purpose of the EXPERIMENT button is to show how cascade failure damage decreases as the network hubs are protected and how the effectiveness of hub protection increases as the structure of the network increases.

FTplus

FTplus.html is the web version, and *FT.jar* is the desktop (stand alone) version of the fault tree analysis program described in the book. This software is described in a separate operations manual, see *Fault Tree.pdf*.

FTplus has a number of EXAMPLES that you can use in class to illustrate the power and versatility of fault tree analysis. Each of these is explained, below.

Shared Bank Account : This example parallels the example given in the text. It should be used to support the book's explanation of fault trees.

Bridges of Koenigsburg : This example furnishes the answer to Chapter 5, exercise #10.

SF Hetch Hetchy : This example supports the example given in Chapter 7.

SCADA : This example parallels the illustration in Chapter 8. Note, you will have to change the STEP value from 1.0 to 0.1, to get the results shown in the book. STEP is the size of the DONOR and RECIPIENT steps used by the organizing principle in the Minimal (Optimal) allocation strategy. You need to take small steps to get greater accuracy, because the budget is 10.5 instead of 10 or 11.

Power Grid : This example supports the illustration in Chapter 9 : Power. This fault tree contains the 6 threats explored by the Red Team. It should also be used to compare the results obtained by Event Matrix analysis.

Naperville Power : This example was created by the Chief of Police of the city of Naperville, IL. It shows how to apply MBVA at a local level. Naperville has substations for distribution, but lacks power generators. Substations are important to municipalities even though their failure may not cause a cascade failure.

Kinder Morgan : This example supports the illustration in Chapter 10 : Energy. It illustrates the concept of combination events. Notice that all vulnerabilities are greater than 50%, which leads to the conclusion that the most likely event is a combination event. Any single event failure –Niland, Colton, or Watson – is less likely to occur than a double or triple combination failure.

San Lewis Rey Telecom : Use this example to answer Discussion Questions #4 and #5, Chapter 11 : Telecommunications.

Web Vulnerability (Fig6) : Use this example to study Figure 6 of Chapter 13: Cyber Threats. This example is interesting because each allocation strategy yields very different answers. There is no single “good answer”.

WestBay.com Loss of Data : This example supports Chapter 14, problem #10. It represents a “sum of all knowledge” exercise that forces students to work through a complete example from beginning to end.

Ohio River Ports : This example illustrates the application of fault tree analysis to river/port systems. It is not covered anywhere in the textbook. It is an “extra”. It also illustrates a “real world” application of MBVA.

Because fault trees can be saved to disk from FT.jar, other examples may be available from the author.

InternetVirus

InternetVirus.html is the web version, and *InternetVirus.jar* is the desktop version of the Internet virus propagation simulation program described in the book. The purpose of this program is to illustrate how network worms spread using epidemic-like contagion in a

random and scale-free network, using “susceptible-infected-died” and SIS (susceptible-infected-susceptible) models.

Inputs and Controls



All of the input fields and control buttons are displayed at the bottom of the screen. The inputs are NODES = number of nodes in the randomly constructed network; EDGES = number of links connecting randomly selected nodes; RATE% = infection rate = probability that a node will contract the infectious worm during spreading; and DURATION = number of time steps that elapse between contraction of the worm and disabling of the node. The idea is to “kill off” the infected node after DURATION time units.

The control buttons are:

SCALE-ON : Convert the random network to a scale-free network using increasing returns (preferential selection). Use this to compare the effects of contagion on a structured network.

REPEL/RANDOM : A display option: REPEL forces nodes to maintain a certain minimum distance from one another, and RANDOM allows nodes to perform a random walk around their current location.

TOSS: A display option used to re-position nodes if/when they “walk off the screen”.

Toss randomizes the location of all nodes.

PROTECT: Select a node and then press this button to harden the node. A protected node cannot be infected, and hence cannot fail. Try protecting a hub and compare the effects of

the contagion on the protected network versus an unprotected network. You should see a dramatic difference – protecting the hub should reduce the damage done by a worm.

SIS-ON : Susceptible-infected-susceptible nodes repeat the cycle of being susceptible, then infected, and then after a delay of **DURATION** time units, recovered or susceptible, again. Nodes are colored – **BLUE** = susceptible; **RED** = Infected; **YELLOW** = Recovered, but not susceptible. **YELLOW** nodes are immune to subsequent waves of infection. When **SIS-ON** is pushed, nodes cycle through **BLUE-RED-BLUE** states. When **SIS-OFF** is true, nodes pass through **BLUE**, **RED**, and end up in a permanent **YELLOW** state.

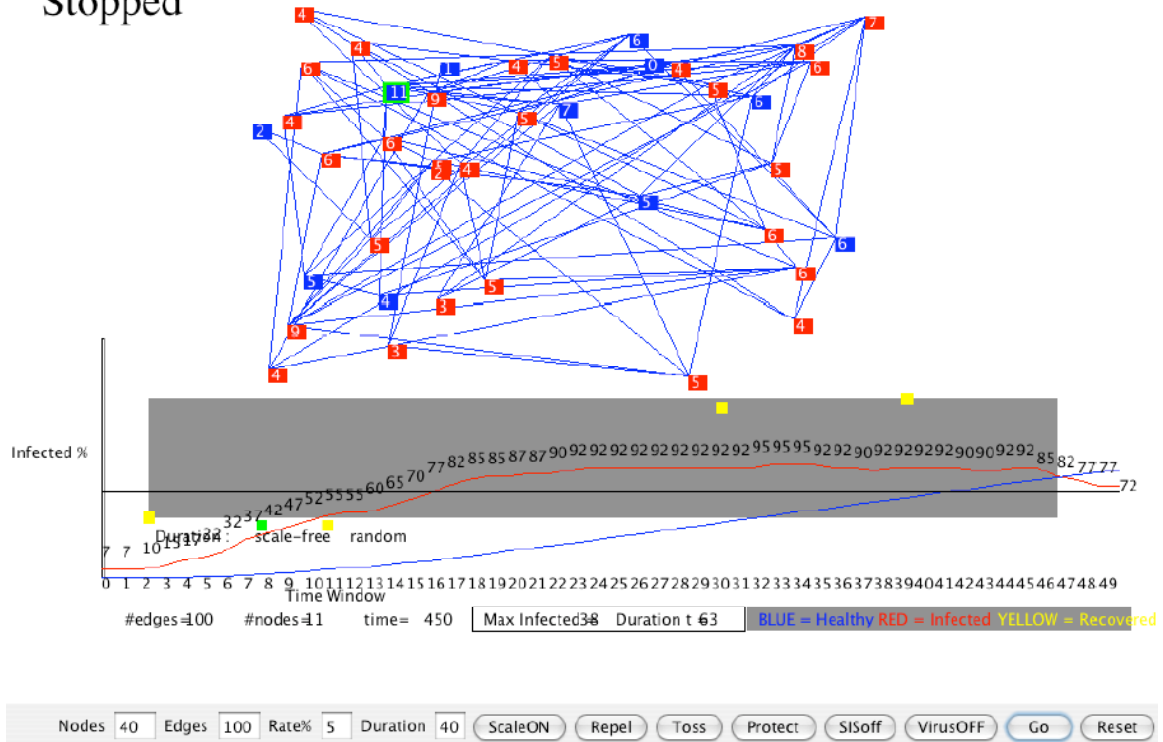
VIRUS-ON : This button initiates the contagion by selecting a node at random and infecting it – coloring it **RED**. The node remains **RED** for **DURATION** time units. Spreading occurs with probability **RATE%**, to adjacent connected nodes. If **SIS-OFF** is true, all infected nodes eventually recover and are colored **YELLOW**. If **SIS-ON** is true, all infected nodes eventually revert to **BLUE**. Subsequent pressing of **VIRUS-ON** causes a subsequent epidemic to begin to spread. If a node is **BLUE**, it can be infected. If it is **YELLOW**, it cannot be subsequently infected. It is immune.

GO/STOP: Always press **GO** to start the program and **STOP** to pause the program. **STOP** does not reset the network. You can alternate between **GO** and **STOP** to view “snapshots” of the animated spread of the virus.

RESET : Pressing **RESET** causes the current network to vanish and a new, random network to be created.

Graphical Output

Stopped



A line graph appears along the lower portion of the screen when VIRUS-ON is pressed. This graph plots the percentage of infected nodes versus time for the last 50 time units of the simulation (and the average value as a horizontal line, and the cumulative value as a blue line). In addition, the maximum number infected and the time it took to reach the maximum are displayed in textual form below the graph.

Students can perform experiments with InternetVirus by running VIRUS-ON many times, using different selections of protected nodes and network structure. The results of these experiments are plotted as small squares – yellow indicates a random network, and green squares indicate the results for a scale-free network. The vertical position of the square corresponds with the maximum percentage of infected nodes, and the horizontal position corresponds with the time to reach this maximum.

Generally, scale-free networks (green) with protected hubs will exhibit lower infection rates than random networks (yellow). This confirms the general results obtained from the study of cascade networks.

NetworkAnalysis

NetworkAnalysis.html is the web-based version and *NA.jar* is the standalone (desktop) version of the Network Analysis program described in the book. *NetworkAnalysis* is one of the major programs used by students, hence it is documented in a separate user manual, see *NetworkAnalysis.pdf*.

NetworkAnalysis contains a number of examples that support the illustrations in the book. Some examples are not included in the book, but are provided for classroom demonstration. In many cases, *FT.jar* contains fault trees corresponding with the network examples in *NA.jar*.

A Small World : The purpose of this example is to illustrate the nature of small world networks. Note how the node frequency histogram is shaped – there are spikes corresponding with clusters. A pure scale-free network has no spikes. Therefore, you can identify a small world network by examining its histogram. Compare this with a network that is “more scale-free” such as the Gulf of Mexico Oil Fields.

Bridges of Koenigsburg : This example illustrates the importance of careful modeling. The Koenigsburg Bridge system is a network of landmasses and bridges. But note how nodes are both landmasses and bridges – not just one or the other. This highlights an important concept: nodes should represent the assets of interest in an infrastructure. So, if your analysis is interested in roads and bridges, then nodes are roads and bridges, and

links are the connectors that link roads to bridges, and bridges to roads. Similarly, the Königsberg Bridges problem is interested in both land masses and bridges, hence land masses and bridges are nodes, and links are the connectors that link land masses to bridges and bridges to land masses.

Example Ch6 : Figure 3 of Chapter 6 is used to illustrate the impact of damage value on allocation. This (small world) network has a hub, which would be identified as the critical node if it were not for the damage values of other nodes. As a consequence of non-uniform damages, allocation favors high-damage nodes over low. Actually, the product of degree times damage value is used to allocate points.

SLR Water Network : This example provides the solution to problem #15, Chapter 7 : Water, and Figure 8. The network is very similar, but simpler, than the Hetch Hetchy water supply network used as a case study.

SLR Gas & Oil Pipeline : This example supports problem #11, Chapter 8: SCADA and corresponds with Figure 10. Students will have to run this example to get the answers to Problem #11.

SFO Water & Power : This example illustrates the example given in Chapter 7 : Water. Note, this example has been extended to include the SF Airport, O'Shaughnessy Dam, and connecting power lines. You will have to trim this network by deleting these extras to get the same network used in the book. You can use this network to show how two different sectors (power and water) are dependent on one another, and therefore interdependent.

Kinder Morgan Pipeline: This example supports the illustration in Chapter 10 : Energy. It also illustrates a subtle concept – the non-uniqueness of allocation. There are many

solutions to this network allocation problem that satisfy the constraints and also produce minimum risk, R . Try allocating a budget of 150 points and you will notice that the program allocates 100 points to Watson, but then any number of points (that add up to 50) will do for Niland and Colton. In other words, there are many solutions. This occurs because the Niland and Colton have equal degrees and equal damage values. The product of degree times damage for Watson is 200, and 150 each for Niland and Colton. Orange and Mission Valley are next with 100, and all other nodes are 50. Hence, the allocation goes to Watson, and then any combination of points that adds up to 50, go to Niland and Colton.

Gulf of Mexico Oil Field : This example supports the case study of the Gulf of Mexico oil field network of Figure 10 of Chapter 10 : Energy.

AEP Power Cluster : This example supports the exploration of the 2003 Blackout and corresponds with Figure 14 of Chapter 9 : Power. You will need to run this example to answer Exercise 22.

Naperville Power : This example illustrates how local government analysts can apply network analysis to their local infrastructure. It is not used in the book, nor is it needed by any of the exercises.

Top Telecom Routes : This network corresponds with the top 30 telecommunication routes in the USA, as given by Figure 9 of Chapter 11 : Telecommunications.

Tier-1 ISPs : This example supports the case study of the tier-1 ISPs described in Chapter 12 : Internet, and Figure 6.

SLR Internet : This example supports the network described in Chapter 13, Exercise 21, and shown in Figure 8. It is not scale-free (nor small world) and the nodes have differing

damage values, and yet, the result of network analysis depends on node degree and damage value. Allocation takes a long time, but in the end, City Hall (node #0) is the critical node – assuming you allocate budget = 100 points – because it has the highest degree times damage value.

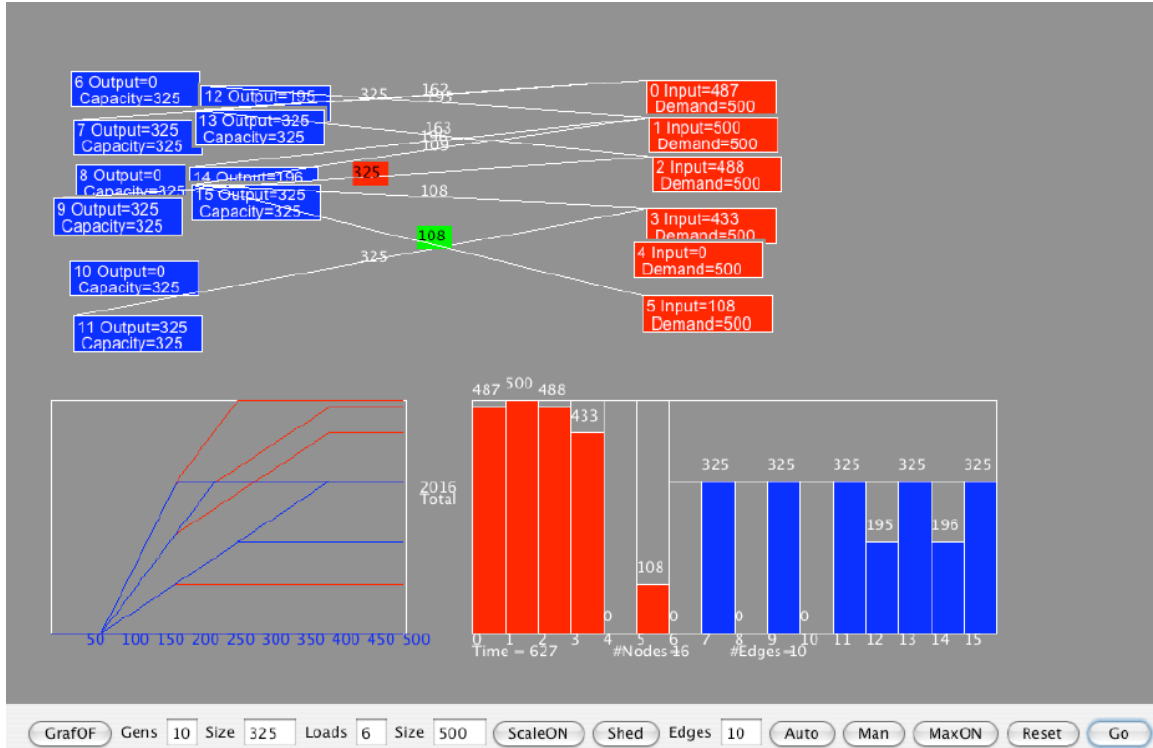
Washington State Ferries : Not discussed in the book, but a good example that illustrates the transportation sector, and a large, realistic network. This is useful for classroom presentation and discussion.

DC Drinking Water : Not discussed in the book, but also a good example from real life. This network was obtained from the WASA engineering diagrams provided by Washington D.C. This example may be more relevant to students of the East Coast than the San Francisco Public Works example used in Chapter 7.

PowerGridSim

PowerGridSim.html is the browser version and *PowerGridSim.jar* is the standalone version of the *PowerGridSim* program described in the book. It implements a highly simplified model of how the power grid works, and is useful for showing students why power level fluctuates due to “buying and selling” through ISOs. Even without the normal fluctuations of demand, power supply level changes when the source of power changes. This fluctuation introduces “vulnerabilities.”

Input Fields and Control Buttons



Reading from left-to-right, the buttons and fields are:

GRAF-ON/OFF: When GRAF-ON is pressed, a graph of power supply versus time appears, and when GRAF-OFF is pressed, the graph disappears. BLUE lines refer to the blue power generators, and RED lines refer to the red consumers or loads. This display is used to watch as power levels fluctuate versus time. The bar charts shown on the right side of the display screen show the same information, but in a bar chart form – one bar for each generator and load.

GENS: The number of generators, initially. Additional generators and loads can be added by pressing the MAN button and entering a value for capacity.

SIZE: This is the default value of capacity used unless overridden by the user. Think of it as the megawatt-hour capacity of a generator or megawatt-hours consumed by a load.

LOADS: This is the number of load boxes, initially. Additional loads can be added later using the MAN button.

SCALE-ON: When this is pressed, the simulator begins “buying and selling” as described by the organizing principle given in the book. Basically, when there is not enough power, lines are switched to other generators in an attempt to increase power. This switching takes time, and consequently, the ACE is non-zero.

SHED: Take the selected load or generator off-line. This can be used to study the impact of load or generator shedding. An off-line generator is colored BLACK.

EDGES: The number of links. The SCALE-ON button tries to maintain a constant number of links equal to this value.

AUTO: Pressing AUTO causes a grid to be created automatically, consisting of the number of generators, loads, and power lines given by the input fields values. First, AUTO creates all loads and generators, and then it randomly connects them with links. Then, power starts to flow from generator to load through links. This flow is gradual until the maximum capacity is reached, or the load demand is satisfied. At some point in time, all flows will stabilize at their maximum values. The largest-capacity link is identified in RED and the smallest-capacity link is identified in GREEN.

MAN: Each time Man is pressed an input dialog will appear requesting either a LOAD or GENERATOR capacity input value. If you have selected a load or generator beforehand, the input dialog value will be used to change the value of the selected box. Otherwise, a new load or generator will be created.

MAX-ON: When this is pressed, the loads that reach their maximum demand remain at the maximum demand level (the red bar chart of these loads remains at the highest-

possible value). This only affects the SCALE-ON organizing principle; hence, it is only useful when searching for the best possible configuration of links-to-generators. This can be used to experiment with ACE control. Can manual interference of the organizing principle beat the automatic optimization of the organizing principle? It appears that PowerGridSim will find the best configuration on its own, given enough time.

RESET: This button stops the simulation and erases the grid. The simulation will have to be started over again by pressing GO and AUTO.

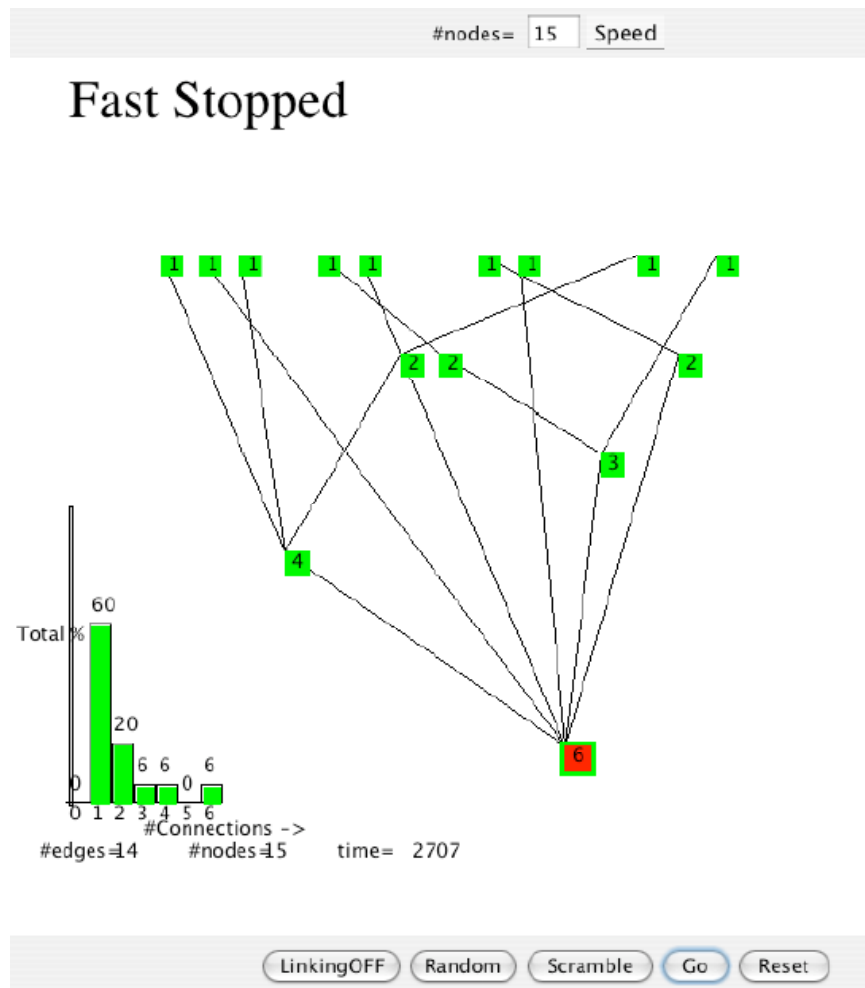
GO/STOP: Start the simulator by pressing GO. When STOP is pressed, the simulation pauses – it does not RESET. Subsequent pressing of GO will resume the simulation.

Note: The built-in grid contains (by default input values) 10 generators and 6 loads connected by 10 links. Initially, these generators deliver 1600 units of power to the loads, but the total demand across all 6 loads is 3,000 units. The total capacity of all generators is 3,250. Hence, it would appear that there is enough power to satisfy all demands, but alas, the shortage of links means that only 1,600 units – approximately 1/2 of the demand – is delivered by the initial configuration. What happens after SCALE-ON is pressed and the simulator has reorganized the links? Is it possible to satisfy the total demand? It appears not. The simulator quickly reaches a configuration that delivers approximately 2,500 units to the load, but then fluctuates up and down around 2400 – 2500. This illustrates the problem with current grid structure in the USA – there are not enough links in the system to deliver all the available power. Thus, a major vulnerability exists in the middle.

RGB (TreeLink)

RGB.html is the web browser version and *RGB.jar* is the stand-alone desktop version of *TreeLink* that is NOT described in the book. Rather, *TreeLink* is the solution to problem #11, Chapter 4. RGB stands for RED GREEN, and BLUE, which is the color scheme used in problem #11. RGB illustrates an organizing principle that ends up forming a tree out of a random network.

Input Fields and Control Buttons



There is only one input field, #Nodes = number of nodes in the network, which appears at the top of the display screen along with a SPEED menu. The SPEED menu allows you to run the program in slow motion so you can study what it is doing.

From left-to-right, the control buttons along the bottom of the display screen:

LINKING-ON/LINKING-OFF: Pressing **LINKING-ON** activates the organizing principle described in problem #11. Blue nodes eventually turn red or green, and eventually only one green node ends up red.

STRETCH/RANDOM: This toggles between **STRETCH**, which forms a tree layout with the largest powered node at the bottom and the lowest powered nodes at the top, and everything in between, well, in between. **RANDOM** scatters the nodes at random.

SCRAMBLE: Each time this button is pressed, the location of all nodes is randomized. Use this to bring stray nodes back onto the screen, if they wander off.

GO/STOP: Press **GO** to start the program, and press **STOP** to pause it. **STOP** will not destroy the network, but **RESET** will.

RESET: Each time **RESET** is pressed, the current network is erased and a new network created. The newly created network has no links. Press **LINKING-ON** to start adding links.

Bar Graph

The bar graph shown in the lower left-hand portion of the display screen is a node frequency histogram – exactly the same histogram used to determine the structure of a network. It shows the percentage of nodes with degree $\#Connections$. Note that this distribution will look very similar to a power law distribution. Is a tree-structured network a scale-free network?

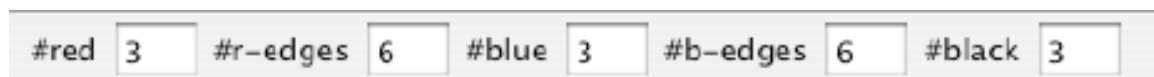
RNet

RNet.html is the web browser version and *RNet.jar* is the stand-alone desktop version of program *RNet* that is described in Chapter 9: Power. It is a special version of

NetworkAnalysis used to analyze tree-structured networks. Most power distribution systems are tree-structured, so why not study the behavior of such networks? *RNet* illustrates how the “middle” of the grid should be protected.

A grid in *RNet* consists of generators (RED), substations (BLACK), and loads (BLUE). The red generators are connected to substations by red links (power lines), and the black substations are connected to blue loads by blue links (distribution lines). Generators have maximum generation capacity; links have maximum capacity, too. But, loads consume whatever power is supplied to them. The objective of *RNet* is to maximize the sum total of power delivered to the loads. This is called *service level*. We want to maximize service level. The problem is, service is diminished when the availability of a generator or substation is less than 100%. Assuming links are 100% reliable, how important are generators and substations? Should generators or substations be protected the most?

Grid Input Fields



A horizontal row of five input fields. Each field consists of a label followed by a text box containing a number. The labels and values are: #red 3, #r-edges 6, #blue 3, #b-edges 6, and #black 3.

The input fields at the top of the display screen are used to define grid components. RED nodes are generators, #r-edges = number of links between red generators and black substations; and BLACK nodes are substations, #b-edges = number of links between substations and loads. The number of loads is specified by #blue.

More Inputs and Control Buttons



A horizontal row of input fields and buttons. It starts with three input fields: RedCap= 100, LinkCap= 100, and R= 4. This is followed by five buttons: Toss, MaxON, Edit, Reset, and Go. The Go button is highlighted with a blue border.

The row of inputs and buttons along the bottom of the screen specify the capacities of power generators and links.

REDCAP: The capacity of each red generator. This value can be changed by selecting the generator and pressing EDIT.

LINKCAP: Transmission capacity of the links. This is a default value; it can be changed by selecting a link and pressing EDIT.

R: R is the total number of points to be allocated to generators and substations during the MAX-ON processing step. An $R = 4$ means 400%, or 400 points to be allocated.

TOSS: Each time you press this button, the network is repositioned on the screen. Use this to return stray boxes to the screen, if they wander off. You can also click on a box and drag it.

MAX-ON/MAX-OFF: This button processes the network according to the organizing principle described in the book. Over time, the points will migrate to the substations, thus proving that grid vulnerability is in the middle. Substations are more critical than power stations!

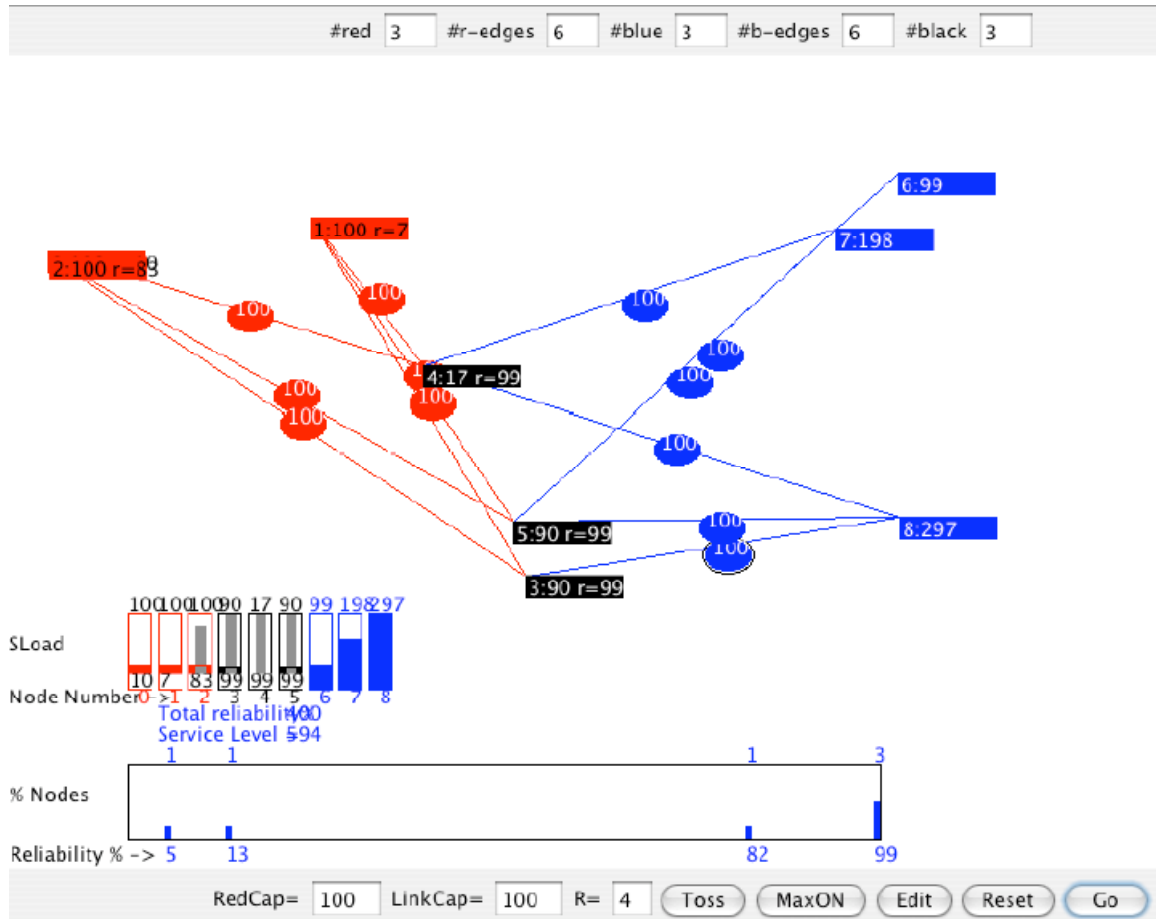
EDIT: Select a link or power generator and press EDIT to change the value of each.

Substations do not have capacities, so they cannot be changed. Loads receive whatever service they receive, so you cannot edit them, either. You can edit only generators and links.

RESET: This button erases the current network and creates a new one. It does not stop a running program, but merely re-creates the simulated grid on the fly. However, it does stop the MAX-ON process.

GO/STOP: Press GO to start the simulator. Press STOP to pause the program. The network is not erased; so subsequent pressing of GO simply resumes the simulation.

Bar Graph



The bar graph appearing in the lower left-hand corner of the display screen is a graphical display of the results of MAX-ON. There is a colored bar for each generator, substation, and load. The colored bar represents capacity, and the gray bar represents the current assignment of points to the generator or substation. These assignments add up to R, which is the given “budget”.

Capacities are displayed in units of ‘capacity’, while the allocated (gray) bars are displayed in percentages. The numerical value of each percentage is shown at the bottom of each bar, while capacity values are shown at the top of each bar.


You will notice that the gray values migrate toward the substations, which indicates that substations are assigned the most availability points. When there are not enough points (R is small), generators lose points to substations. This is the point: substations are more important than power generators. This is a consequence of the fact that substations manage more than one generator, whereas generators manage only themselves.

The display below the bar chart is yet another way to show the allocation. This output plots the %Nodes with non-zero point allocations. This way you can see how the allocation of R tends to cluster around substations.

RSA

RSA.html is the browser version and RSA.jar is the stand-alone version of the RSA program described in Chapter 14. Its purpose is to illustrate the RSA public key algorithm and how it is used to encrypt and decrypt messages.

Input Fields and ENCODE Button



The screenshot shows a web-based interface for the RSA algorithm. At the top, there are three input fields: 'p' with the value '11', 'q' with the value '19', and 'Plaintext' which is currently empty. To the right of these fields is an 'Encode' button. Below the 'Plaintext' field, there are two more input fields: 'Cipher' which contains the text 'null', and 'Inverse' which is empty. The 'Cipher' field has a red label, and the 'Inverse' field has a black label.

The inputs across the top of the display screen are for entering primes p and q required by the algorithm. The PLAINTEXT field is where a user enters a message to be encoded, and the ENCODE button is pressed to cause the encoding. The two fields below this panel are outputs: CYPHER is the encoded message, and INVERSE is the decoded

message. CYPHER is encoded using the public key, and INVERSE is decoded using the private key.

Outputs and Control Buttons



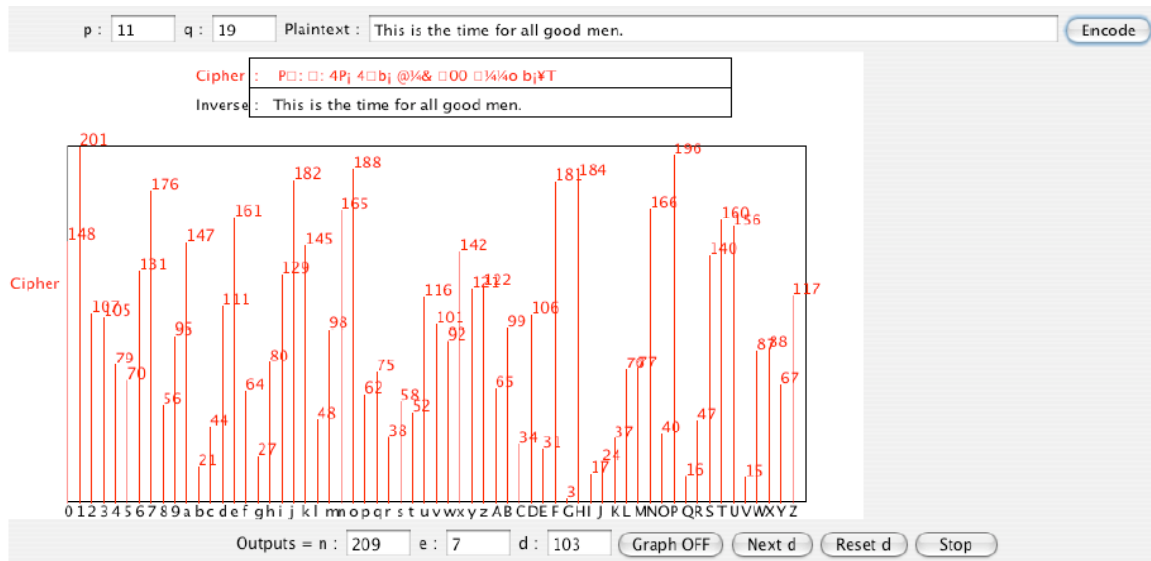
The value of n, e, and d, as described in Chapter 14 are shown, along with control buttons at the bottom of the display screen. Algorithm #2 in Chapter 14 tells how to compute n, e, and d. The control buttons at the right of these fields are for experimentation and exploration.

GRAPH-ON/GRAPH-OFF: This toggle shows/hides a graphical display of the alphabet and digits when encoded. An example of the display is shown below. The point is this: encryption looks like a series of random numbers – there is no discernable pattern. This is good.

Next d: This button causes the algorithm for finding private keys to advance the value of d each time it is pressed.

Reset d: This button resets d to its (initial) lowest value.

GO/STOP: Always press GO first, to start one of these programs. STOP pauses the program, and does NOT reset it.



Rtree

Rtree.html is the web browser version and *Rtree.jar* is the stand-alone desktop version of program *Rtree*. *Rtree* is the tree-structured network equivalent of RNet described in Chapter 9: Power, and needed to solve exercise #21. There is only one generator and one load: everything in between is a tree-structured network of substations. The binary tree of substations has n levels as given by the input value #Levels. The display screen is shown, below.

Inputs and Buttons

GenCap: The capacity of the generator. This can be changed by selecting a block and pressing the EDIT button.

LinkCap: The capacity of links. This can be changed by selecting a link (click on the oval capacity object), and then pressing EDIT.

#LEVELS: *Rtree* supports binary tree transmission and distribution. There are #Levels in this tree, e.g. if #Levels = 4, there are four levels of the tree, which means there are $2^3 = 8$ substations in the final level.

TOTAL-R: This is the budget to be allocated. For example, if $R = 5$, then 500 availability points are allocated to the generator and substations.

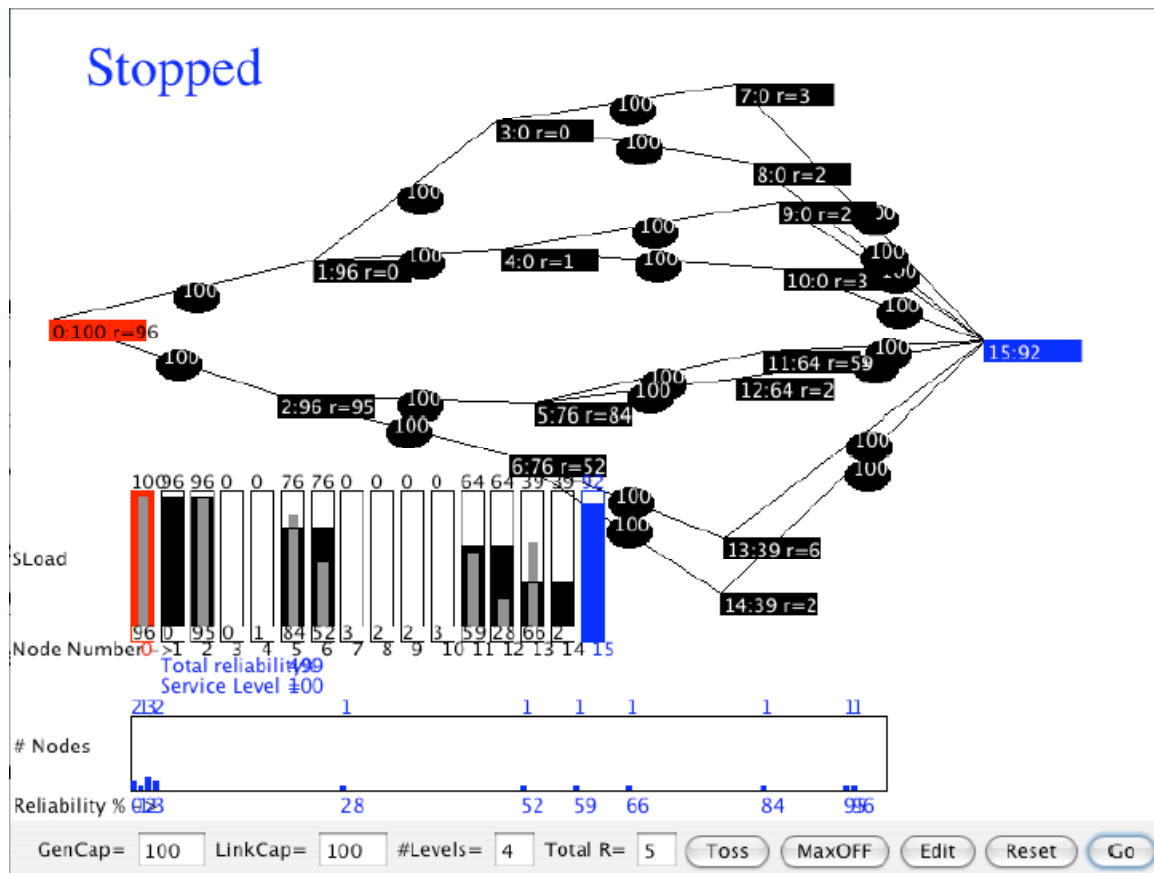
TOSS: Lays out the network in a tree shape. Use this to straighten out the network, especially of nodes crawl off the screen. You can also drag nodes with the mouse.

MAX-ON/MAX-OFF: The allocation algorithm is started with MAX-ON and stopped with MAX-OFF. $100 \cdot R$ points are distributed to nodes such that the amount of power reaching the load is maximized.

EDIT: Select a node or link and press EDIT to change its capacity value. A dialog appears so you can enter a value.

RESET: Resets the allocation to initial (evenly spread) values.

GO/STOP: Always press GO to start; Pressing STOP pauses the program.



Scale-Free(PowerGraph)

PowerGraph.html is the web-based version and *Scale-Free.jar* is the stand-alone version of the scale-free network simulator described in Chapter 4. This simple program illustrates the law of increasing returns (a.k.a. preferential selection), and the power law.

Inputs



There are two inputs and one SPEED menu. #Nodes equals the number of nodes in the randomly created network. #Edges equals to number of links in the network. Links are inserted randomly, per the organizing principle described in Chapter 4: Networks. The SPEED menu is used to run the program in SLOW, MEDIUM, or FAST tempos.

Control Buttons



SCALE-ON/SCALE-OFF: This button toggles the creation of a scale-free network from a random network through increasing returns per the organizing principle described in the book.

REPEL/BROWNIAN: This button is used for display purposes, only. REPEL causes the nodes to attempt to maintain a certain minimum distance apart. BROWNIAN causes the nodes to move randomly around the screen.

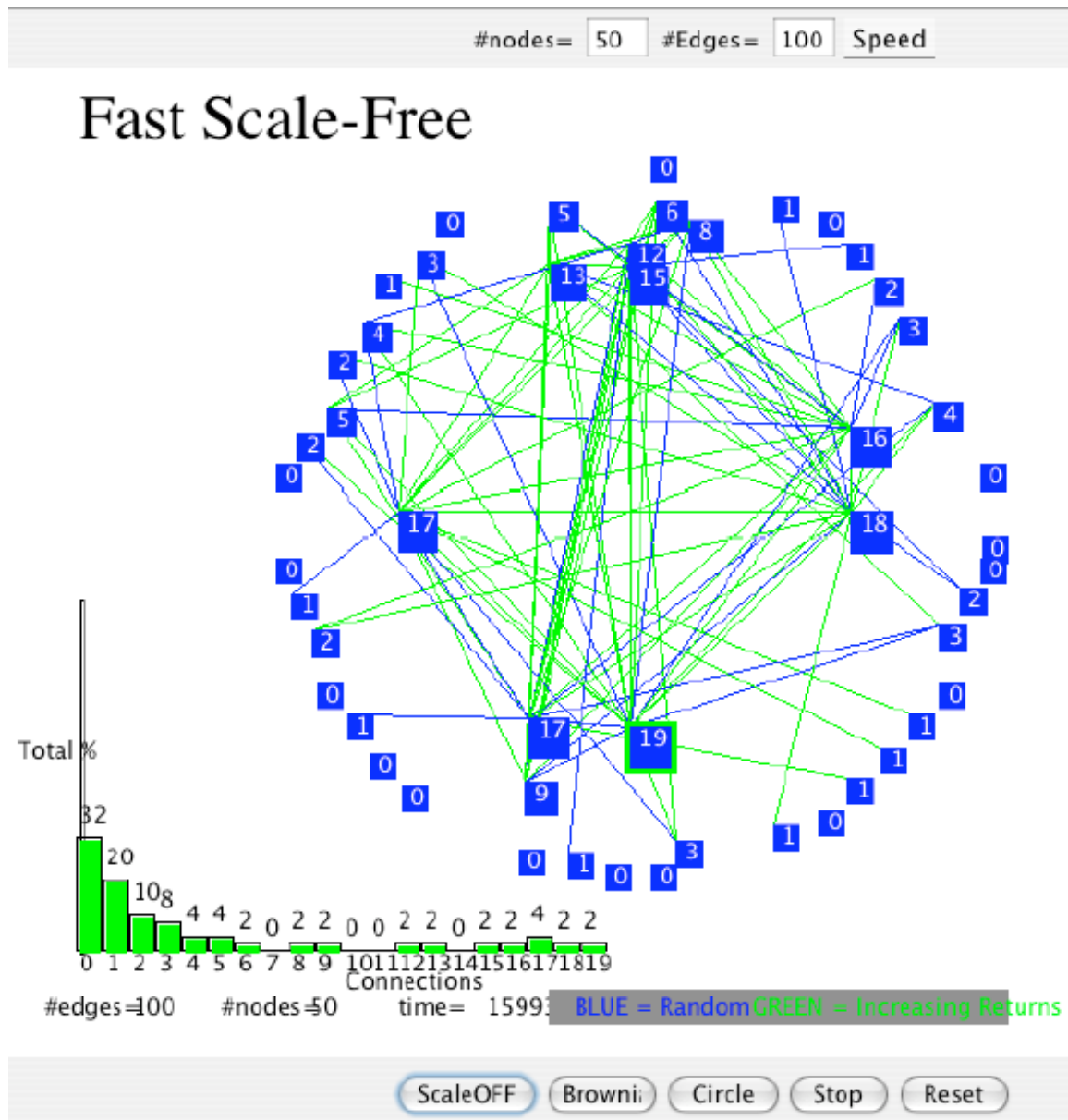
CIRCLE: This button causes the nodes to be arranged around the perimeter of a circle. The distance each node is from the center of the circle is proportional to the degree of the node – the higher the degree, the nearer the center. The edge of the node with the highest degree is colored green.

RESET: Pressing RESET causes the program to reset and start from the beginning:
creating a new network with randomly inserted links.

GO/STOP: Press GO to start the program and press STOP to pause the program.

Display

The display screen contains the network and the node frequency histogram as shown below.



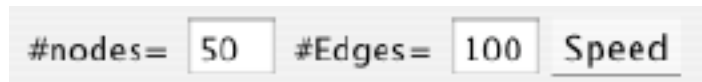
The network is displayed during the process of converting it from a random network into a scale-free network. Links are painted green as they are replaced and the size of the nodes increases as their degree increases. Higher-degreed nodes move toward the center of the circle.

The histogram shown in the lower left-hand corner gradually changes from a Poisson to a Power Law distribution as links are rearranged. The value (percentage) of each histogram bar is displayed along with the height of each histogram bar.

ShortestLink

ShortestLink.html is the browser-based version and *ShortestLink.jar* is the stand-alone version of the *ShortestLink* program described in Chapter 4: Networks. *ShortestLink* is identical to Scale-Free, except the organizing principle is different. *ShortestLink* selects links for replacement according to their length – shorter links are preferred over longer ones. Use it to show that there are other organizing principles besides increasing returns.

Inputs



#nodes= 50 #Edges= 100 Speed

There are two inputs and one SPEED menu. #Nodes equals the number of nodes in the randomly created network. #Edges equals to number of links in the network. Links are inserted randomly, per the organizing principle described in Chapter 4: Networks. The SPEED menu is used to run the program in SLOW, MEDIUM, or FAST tempos.

Control Buttons



ShortenON Repel Circle Go Reset

SHORTEN-ON/SHORTON-OFF: This button toggles the creation of a small world network from a random network through the shortest link organizing principle described in the book.

REPEL/BROWNIAN: This button is used for display purposes, only. REPEL causes the nodes to attempt to maintain a certain minimum distance apart. BROWNIAN causes the nodes to move randomly around the screen.

CIRCLE: This button causes the nodes to be arranged around the perimeter of a circle. The distance each node is from the center of the circle is proportional to the degree of the node – the higher the degree, the nearer the center. The edge of the node with the highest degree is colored green.

RESET: Pressing RESET causes the program to reset and start from the beginning: creating a new network with randomly inserted links.

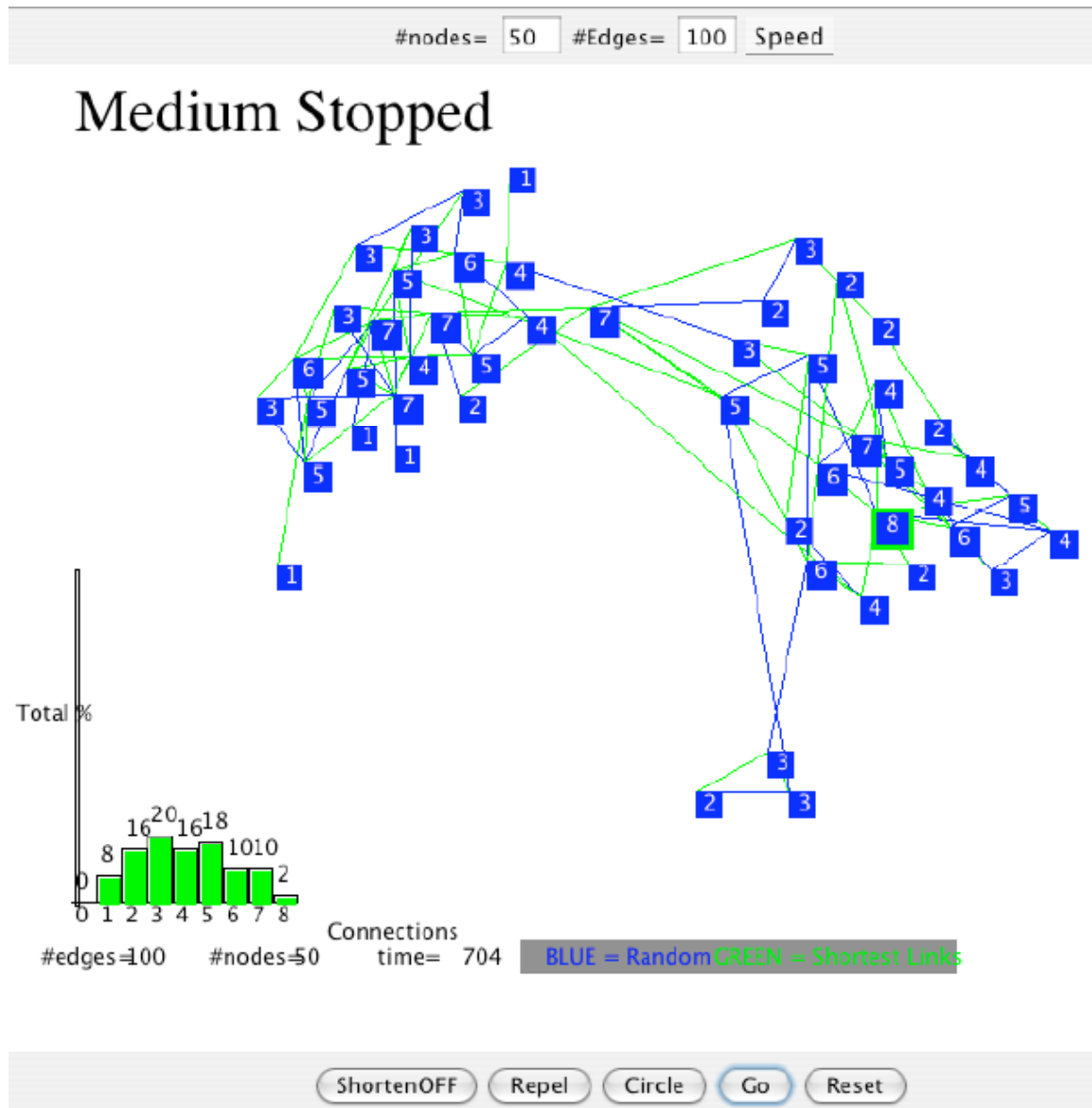
GO/STOP: Press GO to start the program and press STOP to pause the program.

Display

The display screen contains the network and the node frequency histogram as shown below. Note how the network partitions itself into clusters – like a small world network. The shortest-distance organizing principle replaces long links with shorter links, which reinforces the predominance of short links. Perhaps the node frequency histogram should be replaced by a “link length” histogram to show this.

As before, links are painted green when they replace longer links. The histogram is NOT shaped like a power law because the distribution of links to nodes is no longer

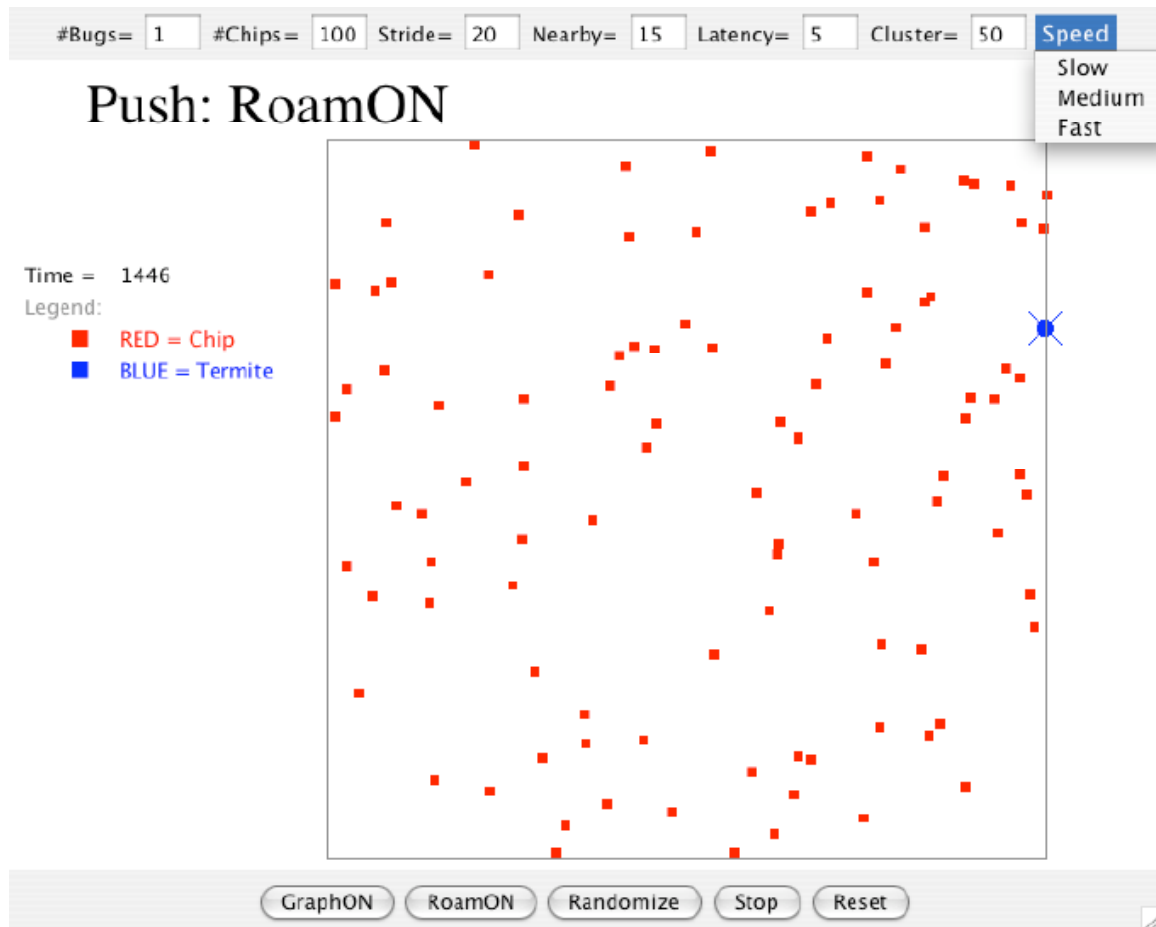
preferential. Rather, replacement of one link by another has little impact on the degree of nodes.



Termites

Termites.html is the browser-based version and *Termites.jar* is the desktop (stand-alone) version of *Termites* as described in Chapter 4: Networks. *Termites* illustrates emergence

and suggests why critical infrastructures tend to cluster or concentrate assets in a small number of locations. The display screen of Termites is shown, below.



Inputs



#BUGS: The number of termites. Setting this to 10 or 20 speeds up the simulation

because there are more termites to do the work.

#Chips: The number of wood chips. Initially, these are scattered at random around the rectangle.

STRIDE: The step size taken by each termite as it randomly roams around the square. Set this to any number greater than 2 and watch as the termites take big (or small) steps.

NEARBY: A termite can pickup a chip when it is within this distance from a chip. This is the radius of the BLUE circle around each chip when GRAPH-ON is set.

LATENCY: The number of steps a termite takes after dropping a chip and before it is able to pick up another chip. If this number is too small, the termite may pickup the same chip repeatedly. Therefore, this is a kind of latency – a dormant period where the termite is not allowed to pickup another chip.

CLUSTER: The red circle around chips when GRAPH-ON is set. This is the radius value used to estimate a cluster. Cluster sizes are computed by counting the number of chips within CLUSTER pixels of one another.

SPEED: The simulation can be run SLOW, MEDIUM, or FAST.

Control Buttons



GRAPH-ON/GRAPH-OFF: GRAPH-ON enables the display of a bar chart and draws circles around clusters. See Below. The bar chart displays percentage of chips in a cluster versus cluster number, where clusters are ranked from largest-to-smallest. A cluster is defined as a circle of radius CLUSTER containing chips. The circles drawn around clusters represent: RED: the radius of the cluster, and BLUE: the radius of the NEARBY or proximity circle. NEARBY is the maximum distance a termite can be from a chip and still pick it up or set one down.

ROAM-ON/ROAM-OFF: Start/Stop the action: this button controls the simulation. When ROAM-ON is pressed, termites perform a random walk and implement the organizing principle described in the book.

RANDOMIZE: Scatter the chips around – at random.

GO/STOP: Start the program by pressing GO, and pause the program by pressing STOP.

RESET: Re-initialize the program – erase the chips and termites and create a new set of chips and termites. Begin, again.

