GridG: Generating Realistic Computational Grids

Dong Lu Peter A. Dinda
{donglu,pdinda}@cs.northwestern.edu
Department of Computer Science, Northwestern University

Abstract

A realistic workload is essential in evaluating middleware for computational grids. One important component of that workload is the raw grid itself: an annotated graph representing the network topology and the hardware and software available on each node and link within it. GridG is an extensible synthetic generator of such graphs that is implemented as a series of transformations on a common graph format. The paper provides a definition of and requirements for grid generation. We then describe the GridG process in two steps: topology generation and annotation. For topology generation, we have both a model and a mechanism. We leverage Tiers, an existing tool commonly used in the networking community, but we extend it to produce graphs that conform to recently discovered power laws of Internet topology. We also contribute to the theory of network topology by pointing out a contradiction between two laws, and proposing a new version of one of them. For annotation, we have developed a mechanism, the *requirements* for a model, and identified the open problem of characterizing the distribution and correlation of hardware and software resources on the network.

1 Introduction

The goal of Grid computing [11, 12] is to give users easy access to arbitrary amounts of computational power within large, wide area distributed computing environments. Middleware such as Globus [10] and Legion [13] simplify using remote resources within a computational grid. To help users find resources, these systems typically provide some form of a grid information service (GIS) such as Globus MDS [4]. Resource monitoring tools such as NWS [23], Remos [5], or RPS [7] can be used to gauge the dynamic availability of found resources.

Designing and evaluating such grid middleware demands realistic workloads. For example, we are in the process of designing and building a grid information service based on the relational data model [6]. In this system, users will be able to pose complex compositional queries that resemble decision support queries. A typical query might look for a group of machines that use the same OS, together have a certain amount of memory, and that the subset of the network connecting them have some bisection bandwidth. To make these queries fast, we implement them using stochastic search, allowing us to trade off between the number of nondeterministically chosen results returned by the query and the amount of work done in support of it. This tradeoff depends strongly on the structure of the grid: the network topology and the characteristics of the hosts, routers, and links within the topology.

While Smith, et al [19] studied the update and query processes on such grid information, there is no extant work and limited available data on the structure of computational grids. We examined the contents of several running GIS systems. The largest dataset we have found, generously provided by Smith, contains fewer than one thousand nodes. Given the limited data sets, a synthetic grid generator is a necessity. Furthermore, even as more data becomes available, it will continue to be useful to have a parametric source of grids. No such generator currently exists.

In response to the need for large datasets, we have built GridG, a grid generator. Our definition of a grid is an annotated directed graph in which the nodes represent hosts, routers, switches, and hubs, and the edges represent network links. The graph is a thus a network topology that extends to the level of hosts. In addition, each node or edge is annotated with information relevant to its use as a part of a computational grid. A grid generator, such as GridG, produces a grid of a given number of hosts. It must meet the following requirements:

- It must produce a realistic network topology. Much is known about the properties of real network topologies: they are connected, and they have hierarchical structures. Furthermore, wide area network topologies, including the Internet, have been recently been found to follow certain topological power laws [9]. A good generator will provide both structure and follow the power laws [20].
- It must generate realistic annotations for hosts and

Effort sponsored by the National Science Foundation under Grants ANI-0093221, ACI-0112891, and EIA-0130869. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation (NSF).

network components. For a host, it should at least provide the architecture type, processor type and speed, number of processors, memory size, disk size, hardware vendor, operating system, and available software. For a link, it should provide the hardware bandwidth and latency. For routers and switches, it should specify the aggregate backplane or switching fabric throughput and latency. It should capture correlations between different attributes (for example, we might expect that memory size increases with processor speed), and between nearby components (for example, a high speed router is unlikely to be connected only to a few slow links).

The networking community has produced a wide range of topology generators. These generators either meet the structure requirement or the power-law requirement. GridG starts with the output of a structure-oriented topology generator (we currently use Tiers [8, 3]) and adds redundancy to it in such a way as to make it conform to the power laws. As far as we are aware, this makes it the first topology generator that provides structured topologies that obey the power laws.

GridG in fact directly enforces only one power law, the so-called outdegree exponent power law. Its outputs, however, show obedience to the other laws as well. In studying the unreasonable effectiveness of the outdegree law, we discovered a new fact: it is either the case that the so-called rank exponent power law is not actually a power law, or that it is more significant than the others. We believe that the former is actually the case, but that over the typical range that is considered, a power law is a useful approximation.

GridG provides mechanisms for annotating each node and edge. These mechanisms are currently based on user-supplied empirical distributions and correlations among different attributes on the same graph element. We are presently developing a mechanism to support correlation among attributes on nearby graph elements. We use the Tiers structural model to give links reasonable bandwidths given their location on the hierarchy.

Very little is known about the characteristics of a grid or network that are represented by the annotations. We point to the information that we think is necessary to develop models of these little studied network and host characteristics. We also describe the various methods, all unsuccessful at this point, that we have considered for acquiring the data to develop such models. The successful collection of this kind of data, and the models that could be developed from it are are a very exciting research opportunity.

In the following, we begin by presenting the overall architecture of GridG in Section 2. While GridG can apply any number of transformations to produce a grid, there are two core steps: topology generation and annotation. In Section 3, we describe how topology generation is done and demonstrate that GridG conforms to the power laws of Internet topology. Section 4 discusses our insights into those laws, including the apparent contradiction between two of them. Section 5 describes the GridG mechanisms for annotation, while Section 6 outlines the requirements of a model for annotation, describes the open research questions posed by the need for such a model and how we are attempting to address them. Finally, Section 7 concludes the paper.

2 Architecture

GridG is implemented as a sequence of transformations on a text-based representation of an annotated graph. The transformations are generally written in Perl, although this is not a requirement. Figure 1 illustrates how these transformations are composed to generate a grid. Currently, we begin with a structured graph without redundancy that is generated by the Tiers topology generator. The number of networks at each level of the topology is the primary input. This also indirectly specifies the number of hosts. The first transformation enforces the power laws by adding extra links to the graph. The outdegree exponent is main input. The next transformation annotates the graph according to user-defined empirical distributions on and correlations over attributes such as memory and CPU. Additional transformations can be added. For example, we can add clusters to sites on the grid. The final output can then be visualized with DOT, used for GIS evaluation, or for other purposes.

3 Topology

GridG generates topologies comprised of hosts, routers, and IP layer links. In GridG's graphs, nodes in WAN and MAN are routers while nodes in LAN are hosts. Routers have switching capability and several IP interfaces while hosts have computing and storage resources. Grids are embedded in the Internet topology and thus should follow its rules. In particular, GridG produces grid topologies that follow the power laws of Internet topology [9], laws that hold true in many kinds of natural and artificial networks [2].

3.1 Power laws of Internet topology

Faloutsos, et al [9] identified three power laws and one approximation in their influential 1999 paper. Figure 2 summarizes these laws. (Figure 3 summarizes the symbols used in this paper.) The rank exponent law says that the outdegree, d_v, of a node v, is proportional to the rank of the node, r_v, raised to the power of a constant R. In

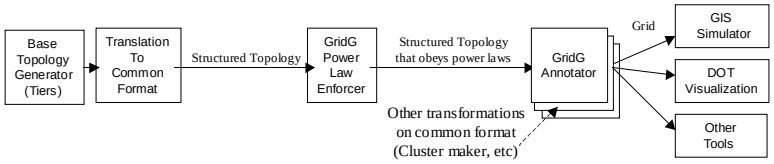

Figure 1: GridG Architecture.

Symbol	Description	Typical Values or Constraints
R	Rank exponent	≈ -0.488 at router level
O	Outdegree exponent	≈ -2.487 at router level
ε	Eigen exponent	≈ -0.177 at router level
H	Hop-plot exponent	≈ 2.84 at router level
d_v	Outdegree of node v	$1 \leq d_v \leq MaxD$
$MaxD$	Theoretical maximum outdegree	$MaxD = e^{-\frac{\log \alpha}{O}}$
r_v	Ranking of node v	All nodes with same outdegree have same ranking
f_d	Frequency of outdegree d	Number of nodes with outdegree d, $f_d \geq 1$
h	Number of hops	
λ_i	The i^{th} biggest eigenvalue of the graph	
$P(h)$	Total number of pairs of nodes within h hops	
N	Total number of nodes in the graph	
β	Constant in equation 1	$\approx exp(4.395)$ at router level
α	Constant in equation 3	$\approx exp(8.52)$ at router level

Figure 3: Symbols used in this paper.

Power Laws	Rank exponent	$d_v \propto r_v^R$
	Outdegree exponent	$f_d \propto d^O$
	Eigen exponent	$\lambda_i \propto i^\varepsilon$
Approximation	Hop-plot exponent	$P(h) \propto h^H$

Figure 2: Power laws of Internet topology.

our examples and evaluation, we choose to parameterize our power laws according to the router-level data in the Faloutsos paper. The parameterized rank exponent law is

$$d_v = \beta r_v^R = exp(4.395) * r_v^{-0.49} \quad (1)$$

The omitted constant term does not affect our results and is commonly dropped [1]. Another useful form of the rank exponent power law is

$$r_v = (\frac{d_v}{\beta})^{\frac{1}{R}} \quad (2)$$

The outdegree exponent law says that the frequency, f_d, of an outdegree d, is proportional to the outdegree raised to the power of a constant O. Parameterizing the law using the Faloutsos router-level data, we have

$$f_d = \alpha d^O = exp(8.52) * d^{-2.49} \quad (3)$$

A node's ranking is defined in the following way, conforming with the Faloutsos paper. We do a topological sort of the nodes in decreasing order of outdegree. We then assign ranks according to this ordering and the number of nodes in each equivalence class. All n_1 nodes in the class with largest outdegree are assigned rank $r_v = 1$. All n_2 nodes in the class with the second largest outdegree are assigned rank $r_v = 1 + n_1$. This accumulation continues such that all nodes in the class with the kth largest outdegree are assigned rank $r_v = 1 + n_1 + n_2 + \ldots + n_{k-1}$. For example, if there are 1000 nodes with outdegree larger than 3, and there are 100 nodes with outdegree 3, then the nodes with outdegree 2 will be ranked 1100. All nodes with same outdegree have the same ranking.

The Eigen exponent power law says that the eigenvalues, λ_i, of a graph are proportional to the order, i, raised to the power of a constant ε.

The hop-plot exponent law is listed as an approximation by Faloutsos, et al. It says that the total number of pairs of nodes, $P(h)$, within h hops, is proportional to the number of hops raised to the power of a constant, H.

3.2 Current graph generators

There are mainly three types of topology generators in use: random [21], hierarchical, and degree-based. Debates as to which type is better for Internet graph generation have persisted over a long period of time [20]. Our belief is that a good graph generator should produce a clear hierarchy that also follows the discovered power laws. Hierarchical generators such as Tiers [3, 8] and Transit-Stub [3] can generate a clear hierarchical net-

work, but the graphs don't follow the power laws by nature. The degree-based generators, such as Inet [14, 22], Brite [15], the CMU power law graph generator [18] and PLRG [1], generate graphs that follow the power laws,but have no clear hierarchical structure. The topologies generated by GridG follow the power laws *and* have a clear three-level hierarchy.

3.3 Algorithms

GridG takes the output of a basic graph generated by Tiers as its input. This input graph has no redundant links. GridG adds links to the graph according to the outdegree power law. Hence, the graphs generated by GridG have a clear three-level hierarchical structure and follow the power laws. The following is a more detailed description.

1. Generate a basic graph without any redundant links using Tiers. Tiers itself has several parameters, specifically the number of nodes and networks at each level of hierarchy. Translate the graph into GridG's native format. This basic graph has three levels of hierarchy: WAN, MAN, and LAN. At each level, the nodes are connected by a minimum spanning tree. Each lower level network is connected to one node on the higher level network. The graph is guaranteed to be connected.

2. Assign each node an outdegree at random using the outdegree power law as the distribution. The probability $P(k)$ that a vertex in the network interacts with k other nodes decays as a power law. This probability is scale-free, meaning that we can extend graphs of any size in this manner [2]. Nodes of outdegree one deviate from the power law as described by Faloutsos, et al [9, Figure 6(b)]. We set $f_1 = f_2$ as this is the case for real router level data. Given outdegree $d = 2, 3 \ldots MaxD$, we calculate the corresponding frequencies according to $f_d = exp(8.9)d_v^{-2.486}$, where 8.9 and -2.486 are the defaults for parameters given a configuration file. $N = \sum_{i=1}^{MaxD} f_i$, where N is the total number of nodes in the graph. We then generate a random number x between 1 and N for each node, if $x \leq f_1$, the node is assigned outdegree 1; if $f_1 < x \leq f_1 + f_2$, the node is assigned outdegree 2; if $f_1 + f_2 < x \leq f_1 + f_2 + f_3$, the node is assigned outdegree 3, etc.

3. Calculate the remaining outdegree of each node after taking the links of the minimum spanning tree into consideration.

4. Add redundant links to the graphs by randomly choosing pairs of nodes with remaining outdegree > 0. Nodes at higher levels (e.g., WAN) are given priority over nodes at lower levels (e.g., MAN). Continue to add more redundant links until no pairs of nodes with positive outdegree can be found.

	Internet Routers	GridG	Tiers
Rank exponent	-0.49	-0.51	-0.18
R^2		0.94	0.89
Outdegree exponent	-2.49	-2.63	-3.4
R^2		0.97	0.55
Eigen exponent	-0.18	-0.24	-0.23
R^2		0.97	0.97
Hop-plot exponent	2.84	2.88	1.64
R^2		0.99	0.99

Figure 4: GridG topology generator Evaluation

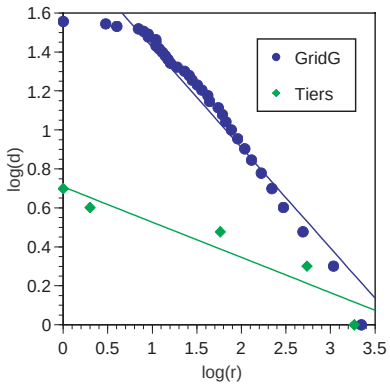

Figure 5: log-log plot of Outdegree vs. Ranking

3.4 Evaluation

In this section, we show that the graphs generated by GridG follow the power laws. For comparison, the basic graph generated by Tiers is also shown in our figures. The basic graph was generated by "Tiers 1 50 10 500 40 5 1 1 1 1 1", meaning one WAN containing 50 MANs, each containing 10 LANs. The WAN contains 500 nodes, while the MANs and LANs contain 40 and 5 nodes, respectively. This is similar to the parameters used in other evaluations [20].

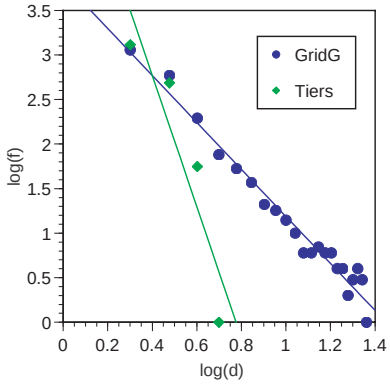

Figure 6: log-log plot of Frequency vs. Outdegree

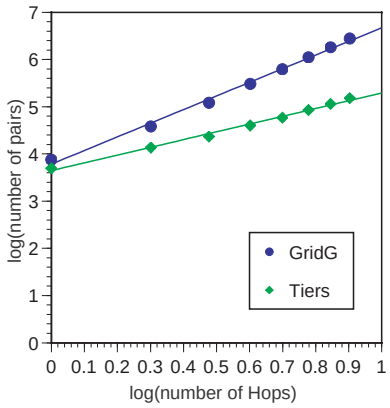

Figure 7: log-log plot of Number of pairs of nodes within h hops vs. Number of hops h

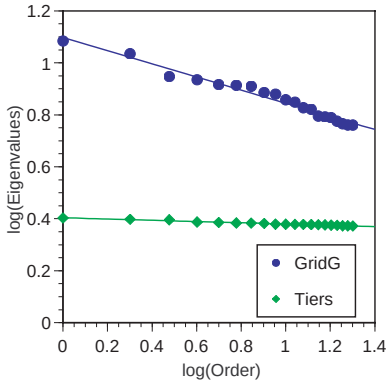

Figure 8: log-log plot of Eigen values in decreasing order

Figure 4 shows that the exponents of the topology generated by GridG match router level data from the Faloutsos paper much better than those of the basic Tiers graph. The coefficients of determination, R^2, represent how well a power law fits the generated data. We can see that GridG produces R^2 values close to 1, the ideal. The exponents in Figure 4 are the slopes in Figures 5, 6, 7, and 8.

Figure 5 is a log-log plot of outdegree versus ranking. We can see that a linear fit on this graph explains the relationship for GridG's topology very well. The divergence at small ranks is quite interesting and shows up in studies of real topologies including Faloutsos, et al [9, Figure 4(b)] and Medina, et al [16, Figure 6]. Removing the three diverging datapoints from Figure 5 increases R^2 to 0.99. In Section 4, we will describe a potential new rank law that models this divergence and can be derived from the outdegree law.

Figure 6 shows a log-log plot of frequency versus outdegree. GridG follows the outdegree exponent power law very well except when outdegree equals 1, which is not plotted in the graphs. We have already noted that this divergence is intentionally induced to better match real topologies.

Figure 7 is a log-log plot of the number of pairs nodes within h hops versus number of hops h. Clearly, GridG's topology conforms to this power law.

Figure 8 is a log-log plot of the eigenvalues in decreasing order. We can see that GridG agrees very well with this power law, though our exponents deviate from the data given by Faloutsos, et al [9].

4 Relationships among power laws

Several recent graph generators [14, 22, 1, 18] and GridG generate graphs according to the outdegree law only. However, the generated graphs follow all four power laws! Why is this possible? A possible reason is that the power laws (Figure 2) are closely interrelated. A recent paper [2] proposed incremental growth and preferential connectivity to explain the phenomenon and origin of the outdegree law. Medina, et al found that the hop and eigenvalue power laws were followed by all the topologies they considered [16]. Mihail and Papadimitriou have shown that the eigenvalue law follows from the outdegree law [17].

In the following, we show that the outdegree law follows from the rank law. It does not appear, although we can not prove, that the rank law follows from the outdegree law. This suggests one of several possibilities:

- The rank law is strictly more descriptive than the outdegree law.
- The rank law is wrong.
- The outdegree law is wrong.

The evidence against the first and third possibilities is the unreasonable effectiveness of using the outdegree law to generate graphs that appear to follow all of the laws. Furthermore, as we noted earlier, the rank/outdegree relationship diverges from a strict power law in actual topologies at small ranks. Finally, earlier work has shown that the eigenvalue law follows from the outdegree law and that most networks exhibit the eigenvalue and hop-plot laws.

Our belief is that the second possibility is the case. We show that it is possible to derive a power law like rank law from the outdegree law that captures the divergence seen in real topologies and gives the appearance of a power law over much of its range. This also would explain the surprising effectiveness of

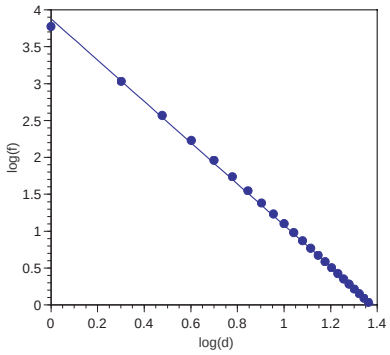

Figure 9: log-log plot of derived f-d law

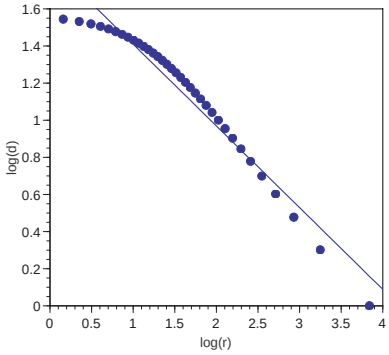

Figure 10: log-log plot of derived d-r law

only using the outdegree law in graph generation. We advocate the following relationship among the laws:

New rank law $\Longleftrightarrow$ Outdegree law $\Longrightarrow$ Eigenvalue law.

4.1 Rank law $\Longrightarrow$ outdegree law

Starting from the rank law, we derive a form of the outdegree law. Let f_d be the frequency of nodes with outdegree equal to d, or the number of nodes with outdegree d. Let r_d be the ranking of the nodes with outdegree d. Similarly, let r_{d-1} be the ranking of the nodes with outdegree equal to $d-1$. Given the outdegrees d and $d-1$, and the ranking of nodes with those outdegrees, the frequency of outdegree d is

$$f_d = r_{d-1} - r_d \tag{4}$$

Now, substitute for r_{d-1} and r_d their values according to the rank law (Equation 2). This gives

$$f_d = (\frac{d_v - 1}{\beta})^{\frac{1}{R}} - (\frac{d_v}{\beta})^{\frac{1}{R}} \tag{5}$$

To simplify further,

$$\boxed{f_d = \beta^{-\frac{1}{R}}[(d_v - 1)^{\frac{1}{R}} - d_v^{\frac{1}{R}}]} \tag{6}$$

This relationship is itself a power law that associates frequency and outdegree. Figure 9 shows the log-log plot of this derived outdegree law (Equation 6). We have derived an outdegree power law from the rank power law.

4.2 Outdegree law $\Longleftrightarrow$ new rank law

Starting from the outdegree law, we attempted to derive a power law for the rank-outdegree relationship. Our end result is a rank law which is not a power law. If our reasoning is correct, then this shows that the rank law can not be derived from the outdegree law. As discussed earlier, we believe that the new rank law, which we derive from the outdegree law, is more accurate than the original rank law in that it fits actual topology data better.

First, note that

$$\sum_{i=1}^{MaxD} f_i = N \tag{7}$$

where N is the total number of nodes in the graph, and $MaxD$ is the maximum outdegree. The minimum frequency is 1, so we must make sure that $f_d = \alpha d^O \geq 1$ (Equation 3). Substituting, we see that

$$MaxD = e^{-\frac{\log \alpha}{O}} \tag{8}$$

From the definitions of rank and frequency, we get

$$r_v = \sum_{i=d_v+1}^{MaxD} f_i \tag{9}$$

That is, the rank of node v is equal to the number of nodes with outdegree bigger than that of node v. Using the outdegree law (Equation 3), we get

$$r_v = \alpha \sum_{i=d_v+1}^{MaxD} {d_i}^O = N - \alpha \sum_{i=1}^{d_v} {d_i}^O \tag{10}$$

If we assume that O is negative, as shown in paper [9], we can then derive the following relationship between rank and outdegree:

$$\boxed{r_v = N - \alpha[\zeta(-O) - \zeta(-O, 1 + d_v)]} \tag{11}$$

Here, $\zeta(t) = \sum_{n=1}^{\infty} \frac{1}{n^t}$ is the Riemann Zeta function.

Figure 10 is a log-log plot of this derived rank law. Surprisingly, this derived law is not an ideal power law—it is far from a straight line. If our derivation is correct, it is clear that the rank law does not follow from the outdegree law. Furthermore, our derived law is a better fit to the actual observed topologies than the rank law. A close look at Figure 10 shows that when rank $r > \approx 37$, the relationship between log rank and log outdegree is nearly

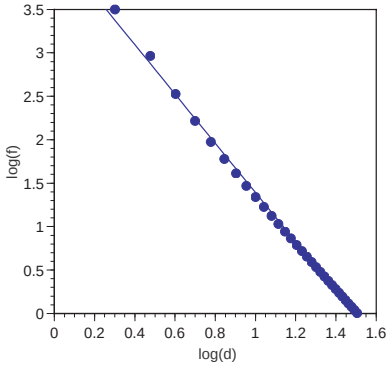

Figure 11: log-log plot of derived d-f law using the new d-r law

a straight line, giving the appearance of a power law relationship. The divergence for $r \leq \approx 37$ is similar to that shown for the actual router level topology in Faloutsos, et al [9, Figure 6(b)].

Figure 11 shows the log-log plot of the outdegree and frequency relationship that can be derived from the new rank law (Equation11) and Equation 4.

5 Annotations

In addition to producing a realistic topology that extends to the level of hosts a grid generator must also annotate the topology with the attributes of its links, routers, switches, and hosts. In GridG we acquire annotations in two ways. For network link bandwidth and latency, we rely on the underlying structural graph generator and power law enforcer. For the switching bandwidth of a router or switch, we assume it is some user-defined fraction of the total bandwidth of incident links.

Host characteristics are considerably more complex. Our first approximation is to treat each attribute of a host independently. The user supplies an empirical distribution for the attribute, and we select randomly based on that distribution. Host attributes are not independent, however. Clearly, host memory and CPU speed are correlated. To express such correlations, our second approximation is to allow the user to specify a joint distribution over all the possible attributes. This is the current state of the GridG annotator.

We envision extending the annotator with a general engine for user-supplied logical rules. For example, the user should be able to assert that hosts with four or more processors have at least 4 GB of memory.

We expect that there exist correlations between the attributes of machines that are near each other in the network topology. The obvious example is a cluster, in which tightly coupled machines have identical attributes. While we can support clusters via an additional graph transformation that explicitly creates them, there are other examples: servers in the same machine room are likely to have more in common with each other than with the client hosts that use them. While we do not yet have a mechanism to support inter-host correlation, we believe that a relaxation algorithm may be appropriate.

6 Open research questions

Given a model of the distribution and correlation of host characteristics, the topology generator and the mechanisms described in the previous section would allow us to generate realistic computational grids. At the present time, we do not have such a model because we have been unable to collect a sufficient amount of data from which to infer one. We appear to be the first to need to do this. In the following, we lay out what is needed, and the approaches that we have undertaken or considered to acquire this data.

We can easily imagine important correlations. For example, the average CPU speed on a Gigabit network is probably faster than on a 10 mbit network. We incorporate this correlation between network speed and CPU speed into our default annotation process. We also assume a positive correlation between the number of CPUs and the total memory, and the assumption that machines with more CPUs are less likely to run a version of Windows.

To test such correlations and to discover more, we need a large random sample of hosts on the Internet or from a representative grid environment. For each host, we need information about its hardware and software resources, and a description of where it is on the network topology captured in the sample. We have considered the following ways of acquiring such a sample:

- Examine the contents of existing grid information service systems. We studied the (anonymized) dumps from several large grids and the dataset collected by Smith, et al [19], which was collected when there was a single MDS server. Smith's dataset was by far the largest, but it contained fewer than 1000 machines. Our conclusion is that existing systems don't contain enough data for our needs.

- Use SNMP scans. The default SNMP MIB provides almost all the information we need. However, most users have turned off SNMP on their hosts due to security concerns. This results in a small and biased sample.

- Use DMI/IPMI/OpenManage/etc. These are BIOS-level distributed management tools for PCs. The prospects here are not clear yet, but even if they could be used, they would provide a sample biased towards PCs.
- Write a virus. An innocuous virus that reported back host data and deleted itself after infecting several other machines would be highly effective, although the sample would be biased towards machines exhibiting the exploits used. We discarded this idea because of its ethical implications.
- Use hardware vendor sales data. If we knew of new machines being sold and attached to the network, we could at least derive distributions of their attributes. We attempted to acquire sales data from IBM, Dell, and HP, but were unsuccessful. These companies regard even aggregated sales data as proprietary information.
- Use OS vendor registration data. Hooking into the processes of OS registration would provide very detailed, if OS-biased information. We attempted to establish a relationship with Microsoft and with Red Hat, but were unable to do so.

7 Conclusion

We have presented GridG, a tool for generating realistic computational grids. The topology generation component of GridG is largely finished. We can produce structured network topologies that obey the power laws of Internet topology. While developing GridG's topology generator, we found that two of the power laws (rank and outdegree exponent laws) are in conflict. We derived a new rank law from the outdegree law that conforms well with published data on actual topologies and has a power law like range. We speculate that this new law is a better approximation of rank behavior.

The topology annotation component of GridG is still at an early stage, but we described the mechanisms we have developed, and, more importantly, pointed out the open research questions that must be answered in order to fully reach the goal of synthetic generation of realistic computational grids. We are now striving to answer these questions.

Acknowledgment

We would like to thank Dr. Huaien Li for discussions on the theoretical part of this paper.

References

[1] W. Aiello, F. Chung, and L. Lu. A random graph model for massive graphs. In *ACM Symposium on Theory of Computing*, 2000.

[2] A. Barabasi and R.Albert. Emergence of scaling in random networks. *Science*, pages 509–512, 1999.

[3] K. L. Calvert and M. B. Doar. Modeling internet topology. *IEEE Communications Magazine*, 1997.

[4] K. Czajkowski, S. Fitzgerald, I. Foster, and C. Kesselman. Grid information services for distributed resource sharing. In *10th IEEE Symp. on High Performance Distributed Computing*, 2001.

[5] P. Dinda, T. Gross, R. Karrer, B. Lowekamp, N. Miller, P. Steenkiste, and D. Sutherland. The architecture of the remos system. In *10th IEEE Symp. on High Performance Distributed Computing*, 2001.

[6] P. Dinda and B. Plale. A unified relational approach to grid information services. *Grid Forum Informational Draft GWD-GIS-012-1*, 2001.

[7] P. A. Dinda. Online prediction of the running time of tasks. *Cluster Computing*, 5(3), 2002.

[8] M. B. Doar. A better model for generating test networks. *IEEE GLOBECOM*, 1996.

[9] M. Faloutsos, P. Faloutsos, and C. Faloutsos. On power-law relationships of the internet topology. In *Proceedings of SIGCOMM '99*, 1999.

[10] I. Foster and C. Kesselman. Globus: A metacomputing infrastructure toolkit. *The International Journal of Supercomputer Applications and High Performance Computing*, 1996.

[11] I. Foster and C. Kesselman, editors. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann, 1999.

[12] I. Foster, C. Kesselman, and S. Tuecke. The anatomy of the grid. *Intl J. Supercomuter Applications*, 2001.

[13] A. S. Grimshaw, W. A. Wulf, and C. T. L. Team. The legion vision of a worldwide virtual computer. *Communications of the ACM*, 40:39–45, 1997.

[14] C. Jin, Q. Chen, and S. Jamin. Inet: Internet topology generator. Technical report, Department of EECS, University of Michigan Ann Arbor, 2000.

[15] A. Medina, A. Lakhina, I. Matta, J. B. famedina, and anukool. Brite: An approach to universal topology generation. In *IEEE MASCOTS '01 (Tools track)*, 2001.

[16] A. Medina, I. Matta, and J. Byers. On the origin of power laws in internet topologies. *ACM SIGCOMM Computer Communication Review (CCR)*, 30:18–28, 2000.

[17] M. Mihail and C. Papadimitriou. On the eigenvalue power law. *Springer-Verlag Lecture Notes in Computer Science*, 2002.

[18] C. R. Palmer and J. G. Steffan. Generating network topologies that obey power laws. In *GLOBECOM '2000*, 2000.

[19] W. Smith, A. Waheed, D. Meyers, and J. C. Yan. An evaluation of alternative designs for a grid information service. *Cluster Computing*, 4:29–37, 2001.

[20] H. Tangmunarunkit, R. Govindan, and S. Jamin. Network topology generators: degree-based vs. structural. In *Proceedings of SIGCOMM '02*, 2002.

[21] B. Waxman. Routing of multipoint connections. *IEEE J. of Selected Areas in Communications*, 1988.

[22] J. Winick and S. Jamin. Inet-3.0: Internet topology generator. Technical Report CSE-TR-456-02, Department of EECS, University of Michigan Ann Arbor, 2002.

[23] R. Wolski, N. T. Spring, and J. Hayes. The network weather service: A distributed resource performance forecasting service for metacomputing. *Journal of Future Generation Computing Systems*, 1998.