AUTOMATED MODEL GENERATION FOR COMPLEX SYSTEMS

Gregory Provan
Department of Computer Science,
University College Cork, Cork, Ireland
g.provan@cs.ucc.ie

Jun Wang
Department of Computer Science,
University College Cork, Cork, Ireland
jw8@cs.ucc.ie

ABSTRACT
It is critical to use automated generators for synthetic models and data, given the sparsity of benchmark models for empirical analysis and the cost of generating models by hand. We describe an automated generator for benchmark models that is based on using a compositional modeling framework and employs graphical models for the system topology. We propose two novel topological models, and demonstrate their advantages, over existing graphical models, in better capturing the topological and functional properties of a class of real system, discrete circuits. We compare generated models to real systems (drawn from the ISCAS benchmark suite) according to two criteria: topological fidelity and diagnostics efficiency. Based on this comparison we identify parameters necessary for the auto-generated models to generate benchmark diagnosis circuit models with realistic properties.

KEY WORDS:
Modeling (Model Development, Automated Model Generation, Diagnostics).

1 Introduction

Creating benchmark model suites is becoming increasingly important for validating a variety of inference algorithms, in domains including VLSI design [9], process control [4], and bioinformatics [12]. Given the sparsity of benchmark models and the cost of generating models by hand, it is critical to design an automated generator for synthetic models and data to analyse the models.

To satisfy this need, we describe a Complex Systems Model Generator (CoSyMGen), which is based on using a compositional modeling framework and employs graphical models for the system topology. Compositional modeling [8] is the predominant knowledge-based approach to automated model construction. It assumes that a system can be decomposed into a collection of components, each of which can be defined using a functional model. These component models are then integrated into the full system model using a system topology graph, which describes the component interactions.

Our use of automated topology generators overcomes the drawback of hand-generated topologies typical of compositional modeling [8]. We base our automated topology generation on the recent discovery that the topology of virtually all real-world systems, from domains as diverse as World Wide Web, social networks, biological systems and technological systems [2, 6] can be modeled using a graph framework [10]. A range of graph models have been proposed, e.g., [1, 10, 13], which are significant improvements over the random graph models traditionally used for empirical analysis of algorithms, in that they capture the topological properties of realistic systems much better than do random graphs [2]. Until now, most analyses of such models have been confined to the models' global statistical properties (e.g. degree distribution, average shortest connecting paths and clustering coefficients) or the statistics of specific local connectivity patterns (motif)[2]. In contrast, little research has focused on the functionality and corresponding complexity of generated graphs in practical applications.

Further, existing models have been inherently inaccurate, due to discrepancies between the graphical parameters of the real systems and those of the auto-generated graphs. For example, the well-known Watts-Stogatz (WS) [13] model requires an integral mean degree, whereas the mean degree of many systems is non-integral [11].

We address the validity of models generated not only in terms of their topological properties, but also in terms of their functional properties. The functional property that we examine in this article is diagnostics, specifically the inference efficiency of model-based diagnosis (MBD). The MBD problem focuses on isolating the root faults given an observation (e.g., of sensor values).

Our contributions are as follows.

1. We propose a model-generation system, CoSyMGen, which can create models whose parameters can be optimised to conform to a range of different criteria. This tool uses two novel topology-generation algorithms to develop models more accurate than existing approaches.
2. We demonstrate this model generator in the domain of discrete circuits, as an example of real structures that can be investigated using our extended network models, and compare the topological fidelity of the generated models to that of real circuit models.

We organize the remainder of the document as follows. Section 2 compares our contributions to previous results in the literature. Section 3 describes the process we adopt for generating diagnostic models. Section 4 presents the experimental results, and Section 5 summarises our contributions.

2 Related Work

The topology-generation method we adopt was originally developed based on the theory of random graphs–see [2, 10] for background in this area. However, this method focuses solely on the system structure (as captured by the graph), and ignores the system functionality. We extend this approach by adopting the system structure based on the random-graph generators, and then encoding system functionality using a component library.

Compositional modelling uses a set of functional component models, together with a specification of component interactions (called a "scenario" in [8]) to generate useful (mathematical) models. Our approach differs from that of [8] in that we use model generators to create the system structure, or scenario. Further, although the model-generation (or compositional modeling) approach has primarily been applied to physical systems, it can be applied to other domains, such as eco- and bio-systems.

Our model generation approach differs from related work in VLSI auto-generation, e.g., [9], in several ways. The VLSI approach focuses on circuit design for circuit optimisation and simulation; in contrast, our approach can be used for a wider variety of applications, including the diagnostics applications we report. In addition, we adopt random-graph generators for circuit topologies, rather than using optimisation algorithms for defining topologies.

This paper extends the model generation approaches of [11, 12] by using improved topology generation algorithms empirically evaluated over a wider range of circuits.

3 Benchmark Diagnostic Model Generation

This section describes our algorithm for generating benchmark diagnostic models for compositional domains. A domain D is *compositional* if a system model from D can be composed from model components, each of which is defined by a component functional model. CoSyMGen is applicable to any compositional domain for which structural models and component libraries exist.

3.1 Modeling Framework

We assume that a model can be generated from the tuple $(G, \mathcal{F})$, where G denotes the topology graph, and $\mathcal{F}$ denotes the system functionality. The topology graph $G = (V, E)$ consists of vertices V and edges E and specifies the topological relations among the system components. Each node $v \in V$ corresponds to a component or input in the system, and each edge $(v_i, v_j) \in E$ corresponds to a functional relation between v_i and v_j. Our component library specifies a functional description $\mathcal{F}_i$ for each component v_i in the system being modeled.

Generation Algorithm: We generate diagnostic (benchmark) models in a three-step process.

1. generate the (topology) graph G underlying each model;
2. assign components to each node in G for system SD, to create an MBD-graph G';
3. generate the system-level functional model.

We now describe this process using an example, and then describe each step of the process.

Example 1. To demonstrate this approach, we study a suite of auto-generated electronic combinational circuits, which are constructed from simple gates. The inputs to the generation process consist of: (a) a component library of gates (e.g., AND, OR, NOT) together with their functionality (e.g., truth tables, logic equations, probability distributions); (b) parameters defining the system properties, such as the number n of components; and (c) domain-dependent parameters, such as the well-known engineering parameter for determining input/output variables for a circuit, the Rent parameter ξ [3]. Given a desired system with n components, we generate the required topology, as shown in Figure 1(a). The topology of Figure 1(a) depicts the schematic of a simple circuit with arbitrary components A, B, C, D and E. The circuit has two inputs, I_1 and I_2, with the output of component i denoted by O_i. Figure 1(b) shows the circuit with instantiated components. Finally we generate the system functionality in terms of the union of the component functions.

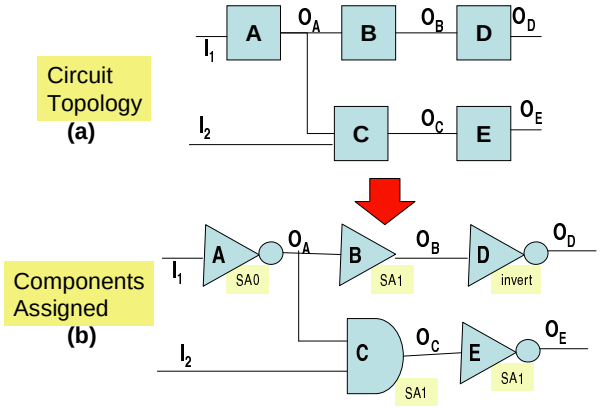

Figure 1. Schematic of simple electronic circuit.

3.2 Graph Topology Generators

We propose two extended topological models that capture the features of real circuits better than existing models: the *extended WS model* and the *extended spatial BA model*.

3.2.1 Extended WS model

The WS approach [13] generates a graph G with a degree of randomness that is controlled by a probability $p \in [0, 1]$. $p \simeq 0$ corresponds to a regular graph, and $p \simeq 1$ corresponds to a random graph; graphs with real-world structure occur in between these extremes, as this methodology has been shown to generate graphs with average shortest path length and clustering coefficient that closely match

real-world systems. Figure 2 depicts the graph generation process, where we control the proportion of random edges using a rewiring probability p. Standard WS generation takes a regular graph (a ring lattice of n nodes), where each node is connected to its k nearest neighbors, and randomly "rewires" an edge by moving one of its ends to a new position chosen at random (with probability p) from the rest of the lattice. We have modified the WS framework to enable us to match the mean degree of the graph for the real system with that of the generated graph, since the mean degree is a critical parameter in this framework. The original WS model requires the mean degree k to be an even number; however, the mean degree in real circuits is typically not an integer. Our enhanced WS generator first creates a ring lattice with the mean degree k, where k is any positive real number, by setting $k' = \lceil \frac{k}{2} \rceil$, and connecting every node to its nearest $\frac{k'}{2}$ neighbours on both sides, just as in the classic WS model. Next it connects every node to its two $\frac{k'}{2} + 1$ nearest nodes on both sides, with probability $\frac{(k-k')}{2}$. Our numerical simulations show that the extended WS model maintains the properties of the classic WS model.

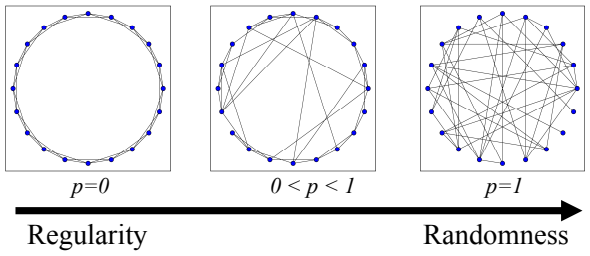

Figure 2. Generating a small-world graph from a regular ring lattice with rewiring probability p.

3.2.2 Extended Spatial BA model

In the standard BA model [1], models are constructed by connecting a new node to m existing nodes in every preferential attachment step, where m is an integer. Our proposed approach is motivated by the observation that networks that occupy 2D space, e.g., road-networks, city-networks, circuits, have inter-node distances governed by a power law [2, 3]. We use a parameter α to impose such a constraint on a standard BA model, to specify a spatial BA model (SBA). We further extend this approach by tuning the mean degree to correspond to that of real systems.

In the SBA model, node position is chosen randomly in 2D space with coordinates in the [0,1] 2D plane. Connections of a new node u with each existing node v are established with probability $P(u,v) \propto k_v l(u,v)^{-\alpha}$, where $l(u,v)$ is the spatial (Euclidean) distance between the node positions, and $\alpha \geq 0$ is a tunable parameter used to adjust spatial constraints and shape the connection probability in the preferential attachment process. When $\alpha = 0$ the model corresponds to the standard BA model.

Similar to the extended WS model, we also extend the preferential attachment process in order to match the mean degree k of the real circuit. In the revised model, a new node is first connected to $m' = \lceil \frac{k}{2} \rceil$ existing nodes by preferential attachment, and we then select the $(m'+1)^{st}$ node by preferential attachment, connecting it with probability $\frac{k}{2} - m'$. Figure 3 displays the spatial BA graph generation by adjusting the geometric constraint.

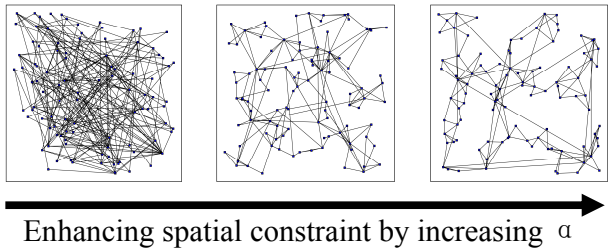

Figure 3. Generating a spatial BA graph by enhancing the spatial constraint.

By continually increasing α, the modularity and clustering of the generated graph will keep increasing, and fewer long-range links will appear. Finally the degree distribution will degrade from the power-law distribution to the exponential or normal distribution.

3.3 Assign Components to graph G

Given a topology graph G, we associate with each node in G a component, based on the number of incoming arcs for the node. Hence, given a node $v \in G$ with i inputs and o outputs, we assign a component, denoted $SD_Z(i, o, \tau, \mathcal{F}_Z, w)$ where τ denotes the type (e.g., AND-gate, OR-gate), $\mathcal{F}_Z$ defines the functionality (behavioural equations) of component Z, and w the weights assigned to variables, e.g., probabilities assigned to the component failure modes of Z.

Example 2. For our experiments, we use a set of m-input gates. Given a node $v \in G$ that has q possible components that are suitable, we randomly assign to v a suitable component with probability $\frac{1}{q}$. For example, the single-input nodes correspond to single-input gates (NOT, buffer), and the dual-input nodes correspond to dual-input gates (AND, OR, NAND, NOR, XOR).

3.4 Generate the System Functional Model

In the final step we generate the system functionality in terms of the union of the component functions, such that we match corresponding inputs and outputs. As an example of input/output matching, consider the following: if output 1 of component X, denoted $O_{X,1}$, is the second input to component Y, denoted $O_{Y,2}$, then we set $O_{X,1} = O_{Y,2}$.

Example 3. For our diagnostic application, given a selected component C_i, we generate the functional description as the union of its normal-mode equations (and potentially its failure-mode equations). We used a probabilistic functional model, structured as a Bayesian Network (BN) [7]. Hence, given the graph G, we assign to each node

$v \in G$ a probability distribution (CPT) $Pr(v|\pi(v))$, where $\pi(v)$ are the parents of v in G.

We randomly select the mode type (of the s possible failure modes) for any component-model with probability $\frac{1}{s}$. We assign a distribution to failure-mode values by assuming that normal behaviour is highly-likely, i.e., $Pr\{C_i = OK\} \simeq 0.99$, and faulty behaviour is unlikely, i.e., $Pr\{C_i \neq OK\} \simeq 0.01$. Figure 1(b) shows a randomly-generated circuit based on the schematic of Figure 1(a). Here, we instantiate components A, D and E to NOT gates, component C to an AND gate, and component B to a buffer. This figure also depicts the instantiated failure-mode for the components in shaded boxes: Components B, C and E have SA1 fault-modes[1], component A has a SA0 fault-mode, and component D has a INVERT fault-mode. Given this information, we can generate a system description with distributions corresponding to the component-types and fault-mode types as just described. For example, the distributions for gates A and C are, respectively, $Pr(O_A|I_1, M_A)$ and $Pr(O_C|I_2, O_A, M_C)$.

3.5 Functionality Parameter Comparison

Given an ISCAS circuit SD with n components, we automatically generate a topology graph underlying the diagnostic model $\overline{SD}$, which has the same node number n and the same mean degree k, by varying the parameter, e.g. the rewiring probability p in the WS model or the geometric constraint α in the SBA model, in order to match the desired property of the real circuit best. We focus on the relative efficiency of MBD inference algorithms on real-world models used to match electronic circuits.

MBD Auto-Generation Task: The objective of MBD auto-generation is to create a model $\overline{SD}$ that minimizes $|\gamma_A(\overline{SD}, OBS) - \gamma_A(SD, OBS)|$, where SD is an MBD model, and A is an MBD inference algorithm that has complexity $\gamma_A(SD, OBS)$ when computing a probability-minimal diagnosis given model SD and observations OBS.[2]

Given the BN approach [7] for our experiments, we used as our measure of inference complexity the largest clique-table in the compiled BN model, which is a typical complexity measure for this type of model. From a theoretical perspective, the complexity of BN inference is expressed in terms of the graph topology: it is exponential in the largest clique of the graph (or the graph width) [7].

4 Experimental Comparison of Generated and ISCAS-Benchmark Circuits

This section summarises experimental results comparing the structure and diagnostic inference complexity properties of auto-generated models with ISCAS benchmark models, which are an established benchmark for circuit optimisation [5]. The benchmark suites consist of multiple sets of circuits, which include the ISCAS85, ISCAS89 and ISCAS99 circuits. We have run experiments for the full suite of ISCAS85 benchmarks. We present only a few demonstrative results here, due to space limitations.

4.1 Comparison of Graph Parameters

This section compares the graph parameters of ISCAS85 circuits with those of models generated by the extended WS and SBA models. The key parameter that we focus on in this article is the degree distribution. To specify this, we first need to define *node degree* in a graph $G = (V, E)$. The degree k_i of a node $v_i \in V$ specifies the number of edges connecting v_i to other nodes in $V \setminus v_i$. The *degree distribution* $p(k)$ of a graph $G = (V, E)$, is a function describing the total number of vertices in a graph with a given degree. The cumulative degree distribution is given by $P(\kappa) = \sum_{k=\kappa}^{\infty} p(k)$. Note that the degree distribution of an Erdos-Renyi random graph follows a Poisson distribution, whereas the models that we are considering have power-law distributions.

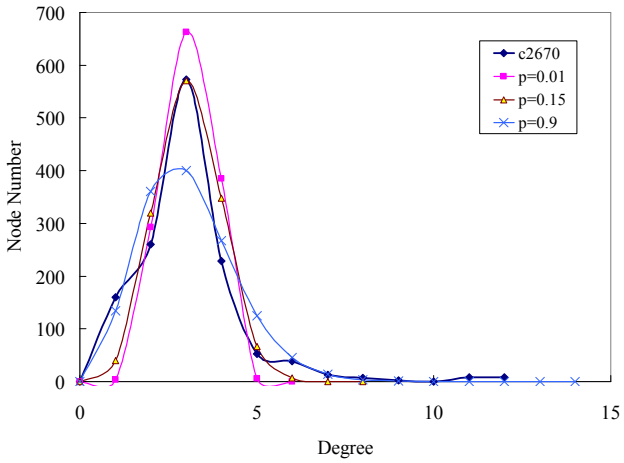

Figure 4. Comparison of circuit degree distribution of C2670 and auto-generated circuits by the extended WS model at various values of p.

The extended WS model is a simple and efficient approach; however, its degree distribution is similar to an exponential distribution, i.e., lacking a long tail. This model is suitable for systems having degree distributions without long tails. Figure 4 compares the degree distributions of ISCAS85 benchmark C2670 and distributions of WS models which have the same node number and mean degree, at various values of rewiring probability p. In general, small circuits can be matched well, since their largest degrees are limited and can be captured accurately by this model.

Compared with the WS model, the spatial BA model has a degree distribution with a longer tail, and better matches real-world degree distributions. This is demon-

[1] A component with a *stuck-at-1*(*stuck-at-0*) fault outputs $t(f)$ (resp.) independent of the input(s).

[2] We assume that $\gamma(\cdot)$ returns a complexity parameter such as CPU-time or number of nodes searched.

strated in Figure 5, which compares the log-log scale cumulative degree distributions of the ISCAS99 circuit B12 and with those of auto-generated circuits with the same number of nodes and mean degree, which were generated by the extended SBA model at different values of α.

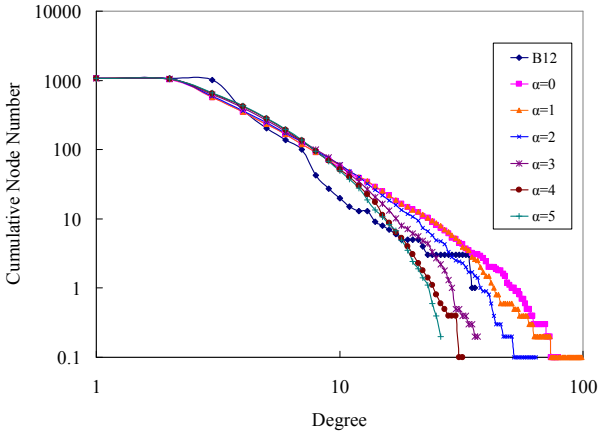

Figure 5. Comparison of circuit cumulative degree distribution of B12 and auto-generated circuits by the extended SBA model at various values of α.

4.2 Diagnosis Complexity of Extended WS model

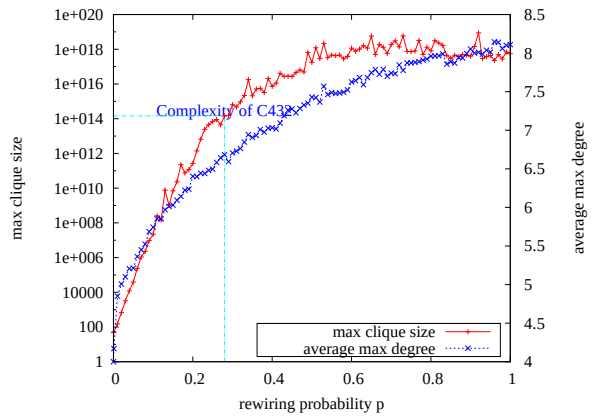

Figure 6. The inference complexity and average maximal degree distributions of a WS model corresponding to benchmark C432.

We ran experiments to explore the complexity of auto-generated WS models corresponding to all ISCAS85 circuits over the entire range of the rewiring parameter p. The data presented is based on an average of 300 runs. Figure 6 shows the average maximal degree of the generated graph G (corresponding to C432) compared with the $\log_{10}$ of the inference complexity of G. The figure shows that both curves have the same trend, rising from the relatively efficient regular graph ($p = 0$) to the range of small-world graphs ($0 < p < 1$). As the shown in Figure 4, increasing p flattens the degree distribution of WS model but increases the length of tails and the degree of the hub nodes (and hence increases the inference complexity). We experimentally selected the rewiring probability p that minimised $|\gamma_A(\overline{SD}) - \gamma_A(SD)|$ over a broad range of observations. For example, empirical comparisons showed that the WS model with $p \simeq 0.28$ produced a model with inference complexity closest to that of C432—see Figure 6. In our experiments we found that the ISCAS85 circuits all had corresponding WS models with $0.01 < p < 0.3$.[3] Note that $0.01 < p < 0.3$ corresponds to WS models of relatively low complexity, i.e., we can generate models of significantly higher complexity than the ISCAS85 circuits, thus providing a range of difficulty of auto-generated models. Analysis of all ISCAS85 benchmark circuits shows similar results,; also, note that the WS model can only vary the length of the tail of degree distribution in a limited range.

If we use the classic WS model, which requires k to be even, the closest k for matching C432 is 4, causing the degree distribution, and hence the diagnosis complexity of the generated graph to be quite different from that of the real circuit.

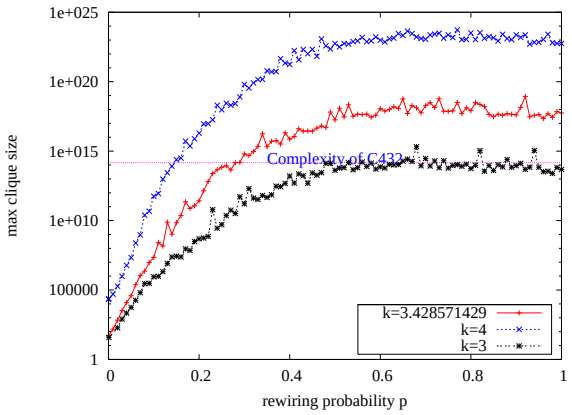

Figure 7. The inference complexity of WS models with the same node number but different values of k

4.3 Diagnosis Complexity of Extended SBA model

Another set of experiments explored the complexity of auto-generated SBA models corresponding to all ISCAS85 circuits over a broad range of the parameter α, which is used to tune the spatial constraint. Since the complexity of generating the SBA model is much higher than that of the WS model, to ensure the problems were of manageable size for the current inference algorithm, the data in this set of experiments are based on an average of 100 runs.

Figure 8 shows the average maximal degree of the generated graph G (corresponding to C432) compared with a $\log_{10}$-scaled curve of the inference complexity of G. The figure shows that both curves decline with increase in α, as do all ISCAS85 circuits.

[3] It is interesting to note that our auto-generated circuits had an effective Rent parameter similar to that of the original circuits, indicating the correctness of the approach.

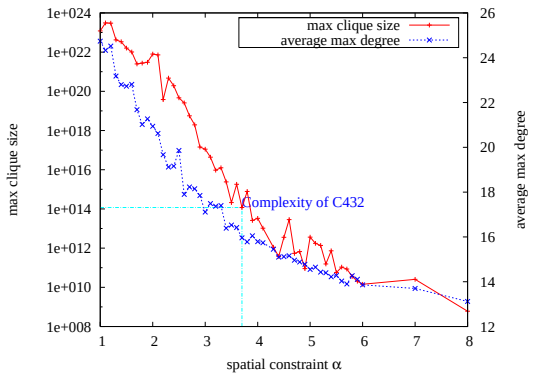

Figure 8. The inference complexity and average maximal degree distributions of a spatial BA graph corresponding to benchmark C432.

The results also show that the hub nodes with high degree affect the diagnosis complexity. Since increasing the spatial constraint results in sharper cutoffs in the tails of the degree distributions, as shown in figure 5, increasing α will limit the tail length in the degree distribution, as well as the degree of the hub nodes (and hence reduce the inference complexity). We again experimentally selected the parameter α that minimised $|\gamma_A(\overline{SD}) - \gamma_A(SD)|$. Figure 8 shows that the generated graph with $\alpha \simeq 3.7$ produced a model with inference complexity closest to that of C432. Similar to the WS model, we can generate benchmark models with a broad range of complexity by tuning the parameter, but the SBA model can adjust the length of the tail of the degree distribution in a much wider range.

The standard BA model cannot match the real circuit with non-integral mean degree, so we extended it by a method similar to that used to extend the standard WS model, while maintaining the same properties of the BA model. When we generated a model with $\alpha = 0$, which corresponds to the BA model without the spatial constraint and contains a very long tail in its degree distribution, and therefore its inference complexity will be too high to match that of the real circuit. Hence, additional parameter tuning was necessary to identify auto-generated models that match both topological graph parameters and inference complexity of the real models; the tendency was for the inference complexity of the auto-generated models to be higher than that of the real models.

5 Conclusion

This article has described a tool, CoSyMGen, for generating functional models that have real-world topology. We have demonstrated CoSyMGen on the domain of diagnostics for circuits. This approach can be used for any domain where systems can be composed from a library of components. For example, if we use a library of pump/engine components, CoSyMGen could build systems from fluid-flow or engine domains. This method circumvents the problems with using random graphs for experiments, and provides an alternative to manually developing suites of benchmark models.

Acknowledgements: Both authors are supported by SFI grant 04/IN3/I524.

References

[1] A.-L. Barabási and R. Albert. Emergence of scaling in random networks. *Science*, 286:509, 2001.

[2] S. Boccaletti, V. Latora, Y. Moreno, M. Chavez, and D.U. Hwang. Complex networks: Structure and dynamics. *Physics Reports*, 424(4-5):175–308, 2006.

[3] P. Christie and D. Stroobandt. The interpretation and application of Rent's rule. *IEEE Tr. VLSI Syst.*, 8(6):639–648, 2000.

[4] D. Dvorak and B. Kuipers. Process monitoring and diagnosis: a model-based approach. *IEEE Expert*, 6(3):67–74, 1991.

[5] J.E. Harlow. Overview of popular benchmark sets. *IEEE Design & Test Comp.*, 17(3):15–17, 2000.

[6] R.F. i Cancho, C. Janssen, and R.V. Sole. Topology of technology graphs: Small world patterns in electronic circuits. *Phys. Rev. E*, 64(4):046119, 2001.

[7] F.V. Jensen, S.L. Lauritzen, and K.G. Olesen. Bayesian Updating in Recursive Graphical Models by Local Computations. *Comp. Stat. Q.*, 4:269–282, 1990.

[8] J. Keppens and Q. Shen. On compositional modelling. *The Knowledge Engineering Review*, 16(02):157–200, 2001.

[9] P.D. Kundarewich and J. Rose. Synthetic circuit generation using clustering and iteration. *IEEE Trans. CAD Integ. Circ. and Systems*, 23(6):869–887, 2004.

[10] M. Newman. The structure and function of complex networks. *SIAM Review*, 45(2):167– 256, 2003.

[11] G. Provan and J. Wang. Evaluating the adequacy of automated benchmark model generators for model-based diagnostic inference. In *Proc. IJCAI-07*, 2007.

[12] T. Van den Bulcke, K. Van Leemput, B. Naudts, P. van Remortel, H. Ma, A. Verschoren, B. De Moor, and K. Marchal. SynTReN: a generator of synthetic gene expression data for design and analysis. *BMC Bioinformatics*, 7(1):43, 2006.

[13] Duncan J. Watts and Steven H. Strogatz. Collective dynamics of "small-world" networks. *Nature*, 393:440–442, 1998.