A Benchmark Diagnostic Model Generation System

Jun Wang, Gregory Provan, *Member, IEEE,*

***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 a three-step process for synthetic model generation: (1) domain analysis; (2) topology generation; and (3) system-level functional model generation. To demonstrate our approach on two highly different domains, we generate models using this process for circuits drawn from the ISCAS benchmark suite and a process-control system. We then analyze the synthetic models 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 and process-control models with realistic properties.**

***Index Terms*—diagnosis, benchmark model generation, compositional modeling**

I. Introduction

Benchmark model suites are vital to facilitating progress in a variety of domains, and the presence of good benchmarks has had big impacts on several areas. For example, the benchmarks for SATISFIABILITY (SAT), e.g. SATLIB[1] and DIMACS[2] have spurred progress in that area; further, it has enabled SAT algorithms to be applied to a variety of other domains, such as planning [1]. Benchmark model suites are becoming increasingly important for validating a variety of algorithms in other domains, including VLSI design [2], [3], process control [4], [5], [6], and bioinformatics [7].

Diagnosis, in contrast to areas such as SATISFIABILITY and constraint satisfaction (CSP), has very few benchmarks. To our knowledge there are only two publicly-available benchmarks for diagnostics, the ISCAS benchmark models for discrete-valued models [8], and the DAMADICS benchmark for continuous-valued models [9]. 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 satisfy the need for benchmark models, we describe a *domain-independent* Complex Systems Model Generator (CoSyMGen), which is based on using a compositional modeling framework and employs graphical models for the system topology. Compositional modeling [10] 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.

Standard compositional modeling tools, e.g., [11], [12], [13], [14] require manual construction of component models, and then use the component library to speed up this tedious manual process of system-level model development. This manual process is necessary to capture target systems, but is costly for compiling a suite of similar benchmark models for tasks like algorithm analysis. Our use of automated topology generators overcomes the drawback of hand-generated topologies typical of compositional modeling [10] by using topology generators for this task.

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 [15], [16] can be modeled using a graph framework [17]. A range of graph models have been proposed, e.g., [18], [17], [19], which are significant improvements over the classic random graph models traditionally used for empirical analysis of algorithms, in that they capture the topological properties of realistic systems much better than do classic random graphs [15]. Although it is known that different domains have different properties, e.g., [20], [15], there has been little work on characterizing domains based on underlying properties. Further, until now, most analyzes 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)[15]. 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 [19] model requires an integral mean degree, whereas the mean degree of many systems is non-integral [21].

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) [22], [23], [24]. The MBD problem focuses on isolating the root faults given an observation (e.g., of sensor values). More formally, MBD determines whether an assignment of failure status to a set of mode-variables is consistent with a system description and an observation (e.g., of sensor values). This problem is addressed in various engineering fields, and the underlying structure

Special Issue on Model-based Diagnosis: Facing Challenges in the real world guest edited by Provan, Struss, deKleer, and Biswas

Jun Wang and Gregory Provan are with the Department of Computer Science, University College Cork, Ireland. e-mail: (jw8/gprovan@cs.ucc.ie).

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

Manuscript received April xx, 2008; revised yy, 2008.

[1]http://www.cs.ubc.ca/ hoos/SATLIB/benchm.html

[2]ftp://dimacs.rutgers.edu/pub/challenge/satisfiability/benchmarks/

of the problem which affects the diagnosis complexity is governed by the graph framework [21].

In this article, we assume we have a library of functional component models for the domain in question, so the main focus of benchmark generation is on creating ensembles of random but "realistic" topologies. A range of methods exist to generate system topologies, each of which has a set of specific input parameters that must be optimized to create a model that accurately depicts a domain-specific topology. As we will show, the different generation methodologies produce quite different models, with different topological properties, such as degree distribution, etc. Since each application domain requires different topological properties, the key to generating good benchmark models is to match the generation methodology to the domain requirements.

Our contributions are as follows.

1) We propose a domain-independent synthetic model-generation system, CoSyMGen, which can create models whose parameters can be optimized to conform to a range of different criteria.
2) We describe the main phases of model-generation:
   a) domain-analysis, which extracts topology and component statistics, and selects fidelity metrics Φ.
   b) topology-generation, which automatically, rather than manually, generates a high-fidelity system topology G by optimizing corresponding parameters Π in terms of Φ.
   c) functional model-generation, which uses the domain library, the component statistics and the system topology (G), to generate the behavioral equations.
3) We illustrate the domain-analysis and model-generation procedure on two quite different domains: (1) model-based diagnosis of discrete Boolean circuits (where we compare the topological fidelity of the generated models to that of real circuit models), and (2) process control diagnostics for a pulp mill.

We organize the remainder of the document as follows. Section II describes the assumptions underlying our model-generation approach. Section III illustrates the system architecture of CoSyMGen. Sections IV, V and VI describe the domain analysis, system topology generation and system functional model generation in the proposed benchmark generation process, respectively. Sections VII and VIII present the experimental results of diagnosis benchmark generation on the two different application domains. Section IX compares our contributions to previous results in the literature. Finally, section X summarizes our contributions.

II. Key Assumptions

This section discusses two key assumptions underlying this approach: model compositionality and the ability to generate realistic system topologies. We first introduce the necessary notation to discuss these topics.

A. Notation

The models used in standard MBD frameworks, such as the qualitative (logical) [22] or FDI [23], [24] approaches typically define a system model in terms of a set $\mathcal{B}$ of behavioral equations, defined over sets of failure-modes and observable/controllable variables. The underlying assumption is that such systems are not explicitly viewed as being decomposable, and the system model is treated monolithically. In contrast to these approaches, we are interested in system models that are explicitly decomposable, so we must capture not just the behavioral equations, but also a framework that captures the system's compositional properties.

A system is compositional if its behavior consists of the combination of the behaviors of its constituent components' behaviors. Due to this assumption, modeling/analysis of compositional models can be more efficient than non-compositional modeling/analysis, and scale better. A precondition for compositional modeling is the existence of an underlying structure for the model. This will describe how the different components of the system constrain each other.

We can thus describe a decomposable model Ψ using two orthogonal aspects: *behavior* and *topology (interaction)*. The behavior model describes the (possibly dynamic) behaviors of the system and components; the topology model describes the component connectivity in terms of components and their connections, and defines the constraints on component behaviors that enable their interactions to specify the system-level interaction [25].

Definition 1. *A composable system Ψ is defined using the pair $(\mathcal{B}, G)$, where $\mathcal{B}$ is the behavior model and G is the topology model.*

We assume that a system Ψ can be decomposed into sub-systems. There are two types of sub-system: a component, which is a primitive sub-system, and a composite sub-system, which can be further decomposed. A component represents the specification of a primitive functionality of Ψ, i.e, no further decomposition of functionality is possible that allows each sub-function to coherently describe a process. We assume a set $C = \{C_1, ..., C_m\}$ of components, and that the input/output tuple for each C_j can be specified. By merging components and/or sub-systems, we can define a hierarchical model; we define a flat model to consist of a system represented only in terms of components $C_i \in C$ and their interconnections.

In this article we focus on two classes of model, a propositional logic MBD model, and a Bayesian network model, as described below.

1) Model-Based Diagnosis Model: This section describes the general MBD framework, which we will use to illustrate our model construction process throughout this article. We can characterize a MBD problem using the triple $\langle$COMPS,SD, OBS$\rangle$ [22], where:

- COMPS= $\{C_1, ..., C_m\}$ describes the operating modes of the set of m components into which the system is decomposed.
- SD, or system description, describes the function of the system. This model encodes the system's topology within the equations in SD.
- OBS, the set of observations, denotes possible sensor measurements, which may be control inputs, outputs or intermediate variable-values.

We adopt a propositional logic framework for our MBD system behavior models SD. Component i has associated mode-variable C_i; C_i can be functioning normally ($[C_i = OK]$), or can take on a finite set of abnormal behaviors.

MBD inference, using *weak fault models* [26], assumes initially that all components are functioning normally: $[C_i = OK]$, $i = 1, ..., m$. Diagnosis is necessary when $SD \cup OBS \cup \{[C_i = OK] | C_i \in COMPS\}$ is proved to be inconsistent. Hypothesizing that component i is faulty means switching from $[C_i = OK]$ to $[C_i \neq OK]$. Given some minimality criterion ω, a (minimal) diagnosis is a (ω-minimal) subset $C' \subseteq COMPS$ such that: $SD \cup OBS \cup \{[C_i = OK] | C_i \in COMPS \setminus C'\} \cup \{[C_i \neq OK] | C_i \in C'\}$ is consistent.

In this article, we adopt a multi-valued propositional logic using standard connectives ($\neg, \vee, \wedge, \Rightarrow$). We denote variable A taking on value α using $[A = \alpha]$. An example equation for a buffer X is $[In = t] \wedge [X = OK] \Rightarrow [Out = t]$.

2) Bayesian Network Diagnosis Model: We can frame a diagnosis problem as a Bayesian network (BN), as done in [27]. Using our $(\mathcal{B}, G)$ framework, in a BN the behavior model $\mathcal{B}$ consists of a set of factorized probability distributions, and the topology model G is a graph.

Definition 2 (Bayesian Network)**.** A Bayesian network is a tuple $(\mathcal{B}, G)$, where G is a directed acyclic graph (DAG), and $\mathcal{B}$ is a set of factorized *probability distributions* constructed from vertices V in G based on the topological structure of G. $\mathcal{B}$ satisfies $Pr(V) = \prod_{i=1}^{n} Pr(v_i | \pi(v_i))$, where $\pi(v_i)$ are the parents of v_i in G.

If we model a diagnosis system as in [27], given an observation OBS, a diagnosis consists of the posterior distribution of the failure modes, i.e., Pr(COMPS|OBS). We typically will select those components with highest probability as the most-likely diagnoses. We can also compute multiple-fault diagnoses within this framework [28].

B. Compositionality Assumption

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.

Since the focus of this article is not on the theory of system compositionality, we will assume the all of the domains to which the synthetic generator will be applied are compositional domains, and refer the reader to the literature for precise expositions of system compositionality. There is a large literature on system compositionality: for example, [29], [30], [10], [31]. In this article, we focus on two forms of behavior equation:

- logic: the compositionality of propositional logic models has been described in [32];
- probability equations: if we model a probability distribution in terms of a graphical model, the compositionality of probability distributions is well-defined [33].

We assume that a model can be generated from the tuple $(G, \mathcal{B})$, where G denotes the topology graph, and $\mathcal{B}$ 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. For any system Ψ that can be decomposed into a set of components, we define the graph $G(V, E)$ for Ψ such that

- the vertices V of G correspond to the components, inputs or outputs of Ψ;
- the edges E of G are such that $(v_i, v_j) \in E$ if (v_i, v_j) correspond to the component pair (C_i, C_j) of Ψ and (C_i, C_j) are coupled.

If the coupling relations indicate directionality, then we assume that all edges of G are directed. Our component library specifies a functional description $\mathcal{B}_i$ for each component v_i in the system being modeled.

1) Compositionality Requirements: When we assume compositionality, we assume that components can be represented in terms of a *block*, which consists of the tuple $(I, O, X, \mathcal{B})$, where a block has two types of ports (I is a set of input ports, O is a set of output ports), X defines the internal variables, and $\mathcal{B}$ is the block's functional description (behavioral equations) defined over (I, O, X). The process of composing a system from components consists of selecting appropriate blocks and connecting the outputs of particular blocks to the inputs of other blocks. The output-input connection is possible if the types of the corresponding ports match [12]. Formal studies of causal block diagrams can be found in [34], [12], [29], [35], [36].

Bond graphs are one modeling framework that explicitly capture a block-composition framework for a broad range of continuous-time physical systems. Cellier [37] describes how to interconnect a set of basic bond graph elements within the object oriented modeling language Dymola.

In this article we focus on discrete-valued systems. As an example of composing such systems, consider the case where we have two simple blocks, B_i and B_j, each with one input and one output. We further assume that the equations for blocks B_i and B_j are $O_i = \phi(I_i, X_i)$ and $O_j = \phi(I_j, X_j)$, respectively. Assume that we connect the output of block B_i, denoted O_i, to the input of block B_j, denoted I_j. By our assumption of compositionality, we can compose the equations of B_i and B_j by equating O_i and I_j, or by renaming the input I_j to the name of the output O_i.

Consider an example where B_i and B_j are both inverters, with modes M_i, M_j respectively, where each mode has values {OK, bad}, and with equations given by:

$$B_i : (M_i = OK) \wedge (I_i \Rightarrow \neg O_i);$$
$$B_j : (M_j = OK) \wedge (I_j \Rightarrow \neg O_j).$$

If we rename I_j to be O_i, then we obtain the composed equations:

$$(M_i = OK) \wedge (I_i \Rightarrow \neg O_i);\ (M_j = OK) \wedge (O_i \Rightarrow \neg O_j).$$

If the system equations are defined by differential equations, then an analogous example on block compositionality can be provided, e.g., [34], [12].

In our model generation, we assume that we can represent the block diagram, which consists of a set of blocks connected by directed arcs, in terms of a directed graph G.

2) Topology Graph: The topology graph G defines the connectivity over the system components. For example, electronic circuits can be viewed as graphs in which nodes are electronic components (such as logic gates in digital circuits) and edges are wires in a broad sense [16]. In gene TRNs, nodes represent genes and edges correspond to regulatory interactions at the transcriptional level between the genes [7], [38].

G can be either directed or undirected, depending on the semantics of the component coupling relations. It is important to note, however, that most compositional modeling frameworks assume directionality, either explicitly [29] or implicitly (i.e., deriving the directionality) [34], [12].

C. Topology Generation Assumption

A second key assumption that we make is that the topology of real-world systems can be captured using a random graph framework. In the past several years several recent theoretical studies and extensive data analyzes have shown that a variety of complex systems, including biological [17], [15], social [17], [15], and technological [16], [17] systems, share a common underlying structure, which is characterized by a class of random graph models. In this structure, the nodes form several loosely connected clusters, every node can be reached from every other by a small number of hops or steps, and the degree distribution P_k, which is the probability of finding a node with k links, displays a heavy tail [17]. Several random-graph models have been proposed to capture the real-world graph properties, such as the Watts-Strogatz (or small-world graph)[19] and the Barabasi-Albert (or preferential attachment) models [18].

Throughout this article, we use this topological property to automatically generate the structure of our diagnosis models.

III. System Architecture

A. Architecture

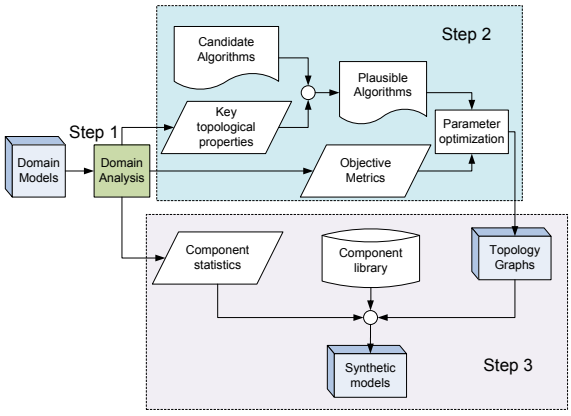

Fig. 1. Automated model generation framework.

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

- **Domain analysis:** analyze existing domain models to extract important model properties;
- **Topology generation:** generate the (topology) graph $\hat{G}$ underlying each synthetic model;
- **System-level functional model generation:** assign components to each node in $\hat{G}$, and create the system-level functional model $\hat{\mathcal{B}}$.

Figure 1 depicts the system architecture. The figure shows how Step 1, domain analysis, computes two types of information: (1) domain topological properties, which are used for topology generation (Step 2), and component statistics, which are used for functional-model generation (Step 3). Step 2 is concerned with generating a system topology graph $\hat{G}$ that captures the domain topology with high fidelity. We have implemented this step in terms of model selection, i.e., selecting the model type (from among a large set of model types that can be created by different model generators) that best matches the topological properties of the original domain model. Finally, in Step 3, using $\hat{G}$, the component statistics and a component library, we create the behavioral model for the synthetic system.

In the following sections, we will show the main model-generation steps in detail.

IV. Domain Analysis Module

We perform domain analysis to capture two types of data:

- component statistical properties.
- topological statistical properties;

We now describe each data type in turn.

A. Component analysis

The analysis of the component properties extracts information about the distribution of components in Ψ, and also potentially the connectivity patterns, or motifs, for the components. The statistical data on components help us to generate more realistic functional models.

For example, for a circuit, we must classify the component types based on their connectivity, and the obtain the relative distribution for each connectivity class. Hence, in a circuit, which has a directed graph as its underlying structure such that every node corresponds to a gate, we identify:

- the set of connectivity classes, where each class is distinguished by the pair η_{ij} =(#-inputs,#-outputs) of every component (node $v \in G$);[3]
- for each η_{ij}, we identify the relative proportion of gates; for example, for η_{11}, we compute the relative proportion of buffers and inverters.

It is possible to generate models using component clusters of size larger than the primitive components for model generation. This approach has been advocated as the proper one for biological domains, among others. For example, various researchers, e.g., [39], [40], [41], have argued that the underlying building blocks of bio-systems, motifs, consist of interacting groups of between 2 and 4 genes, which control transcriptional regulation [42]. Given this evidence, Shen-Orr et al. [40] have

[3]Our current automated procedure deals only with the number of inputs, but in future we plan to extend the implementation to cover both inputs and outputs.

proposed motifs as the basic building blocks in biological and technological networks, and further argue that such motifs possess direct analogues in technological systems. A *motif* has been defined as a subgraph that occurs significantly more frequently in real-world networks than expected by chance alone [43]. The observed over-representation of motifs has been interpreted as a manifestation of functional constraints and design principles that have shaped network architecture at the local level. Some researchers believe that motifs reflect the underlying processes that generated different networks, and may have specific functions as elementary computational circuits in the networks [43]. Motifs may then predict what system-level function a network performs, and how it performs them.

The proposed model-generation approach can be used for synthetic generation based on motifs, as long as the domain analysis is focused on the motif level, rather than the component level.

B. Topological Analysis

We extract topological properties that are widely used to characterize complex systems [44], [45]. The obtained statistical data, based on key topological properties, can help us to select the most plausible topology generation algorithms and behavioral model.

We assume that we have a graph $G(V, E)$ with n vertices and m edges. Some key topological properties are now summarized.

1) Global Graph Properties: Two fundamental global graph properties, which are highly domain-dependent, are characteristic path length $\bar{L}$ and clustering coefficient $\bar{C}$.

Shortest paths play an important role in the transport and communication within a network. For such a reason, shortest paths have also played an important role in the characterization of the internal structure of a graph. A measure of the typical separation between two nodes in the graph is given by the average shortest path length, also known as characteristic path length $\bar{L}$, defined as the mean of geodesic lengths over all couples of nodes.

Graph clustering characterizes the degree of cliquishness of a typical neighborhood (a node's immediately connected neighbors). The clustering coefficient C_i for a vertex v_i is the proportion of links between the vertices within its neighborhood divided by the number of links that could possibly exist between them. The graph clustering coefficient $\bar{C}$ is the average of the clustering coefficient C_i for each vertex v_i [15].

2) Graph Degree Properties: The degree (or connectivity) k_i of a node v_i is the number of edges incident with the node. A list of the node degrees of a graph is called the degree sequence. The most basic topological characterization of a graph G can be obtained in terms of the degree distribution P_k, defined as the probability that a node chosen uniformly at random has degree k or, equivalently, as the fraction of nodes in the graph having degree k.

Degree distributions of some complex systems, such as power grids, appear to have exponential tails: $P_k \propto e^{-k/\kappa}$, as indicated by their approximately straight-line forms on the semi-logarithmic scales [17].

Many real-world systems, such as the WWW and gene transcriptional regulatory networks, are heavy-tailed in their degree distributions. Power laws can characterize their tails, i.e., $P_k \propto k^{-\gamma}$, as indicated by their approximately straight-line forms on the double-logarithmic scales [17]. They are also referred to as scale-free networks, although it is only their degree distributions that are scale-free [17].

Other common forms for degree distributions are power laws with cutoffs [20], [46], [17], such as those seen in electronic circuits and airport networks. The degree distribution looks like a power law over the lower range of values, but decays quickly for higher values. Often, this decay is exponential, and hence this is usually called an exponential cutoff: $P_k \propto k^{-\gamma} e^{-k/\kappa}$, where $e^{-k/\kappa}$ is the exponential cutoff term, and $k^{-\gamma}$ is the power law term.

While most of the focus regarding node degrees has fallen on degree distributions, there are higher-order statistics that could also be considered. [45] introduces the dK-series of probability distributions which specify all degree correlations within d-sized subgraphs of a given graph G. In this framework, the degree distribution P_k is the $1K$ distribution. The $2K$ distribution is the joint degree distribution which describes degree correlations for pairs of connected nodes. The joint degree distribution is

$$P_{k_1,k_2} = \frac{m(k_1,k_2)\mu(k_1,k_2)}{2m},$$

where $m(k_1, k_2)$ is the number of edges between nodes of degree k_1 and k_2, and $\mu(k_1, k_2)$ is 2 if $k_1 = k_2$, and 1 otherwise. The higher-order distributions can be defined analogously. The statistical data of dK distribution can be used the input constraints of random graph generation approach in the subsequent section.

3) Spatial Properties: A particular class of networks are those embedded in the real space, i.e. networks whose nodes occupy a precise position in two or three-dimensional Euclidean space, and whose edges are real physical connections. Along with a complex topological structure, many spatial networks display a large heterogeneity in the wire length of the connections [15]. For example, both electronic circuits and brain networks have heavy-tailed wire length distributions [47], [48].

4) Design Objectives: Various researchers have also proposed optimization approaches as a means of generating system topologies [49], [50]. By investigating plausible objectives and constraints in the design of actual networks, observed topological properties such as node degree distributions can be understood as the natural by-product of an approximately optimal solution to a network design problem. For example, empirical analysis demonstrates that the wire length optimization is among the underlying driving forces creating power law degree distributions with cutoffs in both electronic circuits and brain networks [51], [47], [48].

C. Metrics Selection

We assume that we have a correct set of functional components, meaning that the system topology is the source of model fidelity. In this case, we need to identify metrics for

topology comparison, i.e., methods to measure topological distance $\delta(G, \hat{G})$ between real topology graph G and synthetic topology graph $\hat{G}$.

Naturally, the topological properties discussed in the previous section, i.e., the characteristic path length $\bar{L}$, clustering coefficient $\bar{C}$ and degree distribution P_k can be used as the standard metrics. In addition to these standard metrics, some extended metrics are also introduced for various applications recently [15], [45], [52].

Extended Topology Metrics: We focus on the following extended metrics.

s-Metric: The s-Metric of graph G is defined as $s(G) = \sum_{edge(v_i,v_j)} d_i d_j$, where (v_i, v_j) is the edges in the graph, and d_i and d_j are the degrees of the node v_i and v_j respectively. The s-Metric is closely related to betweenness, degree correlation and graph assortativity [45]. Recent research in both technological and biological systems showed the correlation structure has important impact on system function and performance [53].

Subgraph Frequency Distribution: $P(F_x(G))$ defines that probability of subgraph of type x occurring in graph G. The distribution $P(F_x(G))$ enables us to analyze the frequencies of all sub-graphs with specified sizes, and such subgraph frequency statistics have been successfully applied on evaluation of biological network models [54], [55].

Join-tree Metrics: In many applications involving inference over systems Ψ, e.g., probabilistic inference and model-based diagnosis, the inference complexity has been found to be dependent on parameters of the join-tree $\mathcal{T}$ of the graph G of Ψ [32].[4] As a consequence, for applications involving system inference, we use appropriate join-tree metrics, such as the largest clique size $\mu(\mathcal{T})$ [32], which can be used to represent the inference complexity of the system.

The complexity of inference in a large class of discrete models, which include propositional, CSP and BN models, is exponential in the treewidth of the underlying graph. We can define for a propositional logic model an underlying graph, the topology graph G, which is called the Gaifman graph [56];[5] the constraint graph underlying a CSP is also well-defined in an analogous manner.

We need to select suitable metrics to validate the synthetic topology based on the requirements of the particular application. A parsimonious model can capture the general principles or structures of real-world systems, but it is hard to match all topological metrics simultaneously and perfectly. As a consequence, we need to identify and understand the essential metrics that are responsible for key behaviors of each application, in order to rank the performance of synthetic models primarily in terms of the specified metrics. For example, if the performance of a routing algorithm depends only on the distribution of the shortest path length in the network, then the topologies of a real-world and synthetic network match perfectly as soon as their distance distributions are the same, independent of other characteristics [45], [57].

Our proposed framework enables a user to specify any evaluation metrics. To empirically demonstrate the use of metrics, in this paper focus on generating benchmark models for evaluating the complexity of discrete MBD algorithms, and adopt a join-tree metric that focuses on diagnostic inference complexity, as will be described in Section VII and VIII. We have previously shown that this inference metric more important than other network characteristics [21] metrics.

V. Topology Generation Module

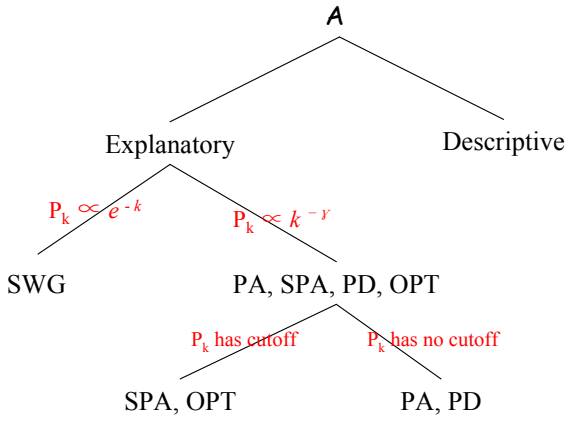

Fig. 2. Selecting the plausible algorithms depending on domain analysis

To generate a synthetic topology graph $\hat{G}$ using an algorithm A, we provide to A a set Π of input parameters. We then measure the properties of $\hat{G}$ (e.g., degree distribution) using a set Φ of graph metrics [45] to compare the properties of the real and synthetic topology graphs, and make adjustments to the generation process, if necessary.

There is a wide range of generation algorithms available for synthetic topology generation, e.g., [15], [58]. Table I classifies the space of topology-generation approaches that our model-generation tool supports, together with their key properties, corresponding parameters, recommended applications, and associated model-generation computational costs. Based on the results of domain analysis, the key properties can be applied to select suitable algorithms. Figure 2 shows an overview of this selection process. For example, the preferential attachment (PA) [15] algorithm requires the number of nodes and edges of $\hat{G}$ as input parameters, and is a viable model when the degree distribution $P_k \propto k^{-\gamma}$ (is power-law distributed) with no cutoff. We classify the generator models into two main groups, as shown in column 1 of Table I: *explanatory* models, which attempt to capture the underlying growth or evolution process of the system topology in the resulting model, or *descriptive* models, which randomly generate topology graphs under the constraints on the specified topological properties, independent of any complex system growth process. For example, the Preferential Attachment (PA) model captures a specific network growth process of complex systems, in

[4]Roughly speaking, the join-tree $\mathcal{T}$ of a graph G is a topological transformation of G into a tree of cliques, where a clique is a fully-connected subgraph [32].

[5]The Gaifman graph of a CNF formula is a graph having a vertex for each variable and an edge (v_1, v_2) if the variables v_1 and v_2 occur in the same clause of the formula. By treewidth (pathwidth) of a CNF formula we refer to the treewidth (pathwidth) of its Gaifman graph.

which new structure preferentially forms around existing substructures [15], and provides a plausible explanation for the origin of the corresponding power law degree distributions. In contrast, given a specified degree sequence, the descriptive generalized random graph (GRG) model [15], [59] randomly forms edges by pairing nodes with uniform probability and satisfies the degree constraint on each node simultaneously. The process we adopt to generating high-fidelity synthetic models differs based on this basic classification. In the following, we will summarize our model selection process (using these two classes), and then review the different generation approaches.

A. Topology Generation using Explanatory Models

Given key properties obtained from topological analysis of real-world network G in domain D, and specified metric set Φ according to domain-specific requirements, we select an explanatory model from a set $\mathcal{A}$ of possible generators (see Table I) as follows:

1) generate potential algorithm set $\mathcal{A}' \subseteq \mathcal{A}$ based on results in domain analysis;
2) optimize parameters Π_i of each algorithm $A_i \in \mathcal{A}'$ to match G in terms of specified topological metrics Φ, and put the A_i into the result set $\hat{\mathcal{A}}$ if it can match G with appropriate values of Π_i;
3) if $\hat{\mathcal{A}}$ contains multiple algorithms, we compute additional metrics Φ', according to further requirements in D, and continue to evaluate and select algorithms in terms of Φ'.

As discussed in Section IV, the degree distribution is one of the most fundamental topological properties, and as shown in Table I, many current topology generation approaches focus on the capability to capture the degree distributions of real-world systems. Figure 2 shows a typical example of the step 1 in the above topology generation process, in which potential algorithms are selected according to analysis on degree distribution. For example, in gene expression simulation, we analyzed the topology graph of the E.coli transcriptional regulatory network (TRN) and found that it displays a clear power law degree distribution. According to Figure 2, the PA and PD model seem viable choices for topology generation. Since the synthetic TRN topology graph $\hat{G}$ are used to generate gene expression data (on which the accuracy of reverse-engineering algorithms is evaluated), we only need to measure the model fidelity in terms of regular topological metrics. For this task, we can use the degree distribution P_k as the basic fidelity metric Φ for the synthetic topology $\hat{G}$. The metric of P_k can be simplified as the corresponding exponent γ when following a power law; the γ of the E.coli TRN is about 2.5. The parameters in the $PA(n, m)$ and $PD(n, m, g_s, p_d)$ algorithms are optimized in terms of γ; n and m are assigned as the numbers of nodes and edges in the actual TRN model respectively. In the step 2, we further optimize values of input parameters to minimize the difference in terms of the γ between the synthesized topology and the actual TRN. The PA model can only generate graphs with γ around 3 deviating from that of the TRN, but the PD model can generate γ in a wide range $(1 \sim 3)$. Finally, the PD model with $(p_d = 0.2)$ closely matches the actual TRN, much better than can the PA model [60]. In some more complicated cases, both the PA and PD model can closely match the degree distribution of a real system, such as the yeast protein interaction network, and we need to compute additional metrics Φ' like subgraph frequency distribution in order to further evaluate and select algorithms, as presented in step 3 [61]. Our topology generation approaches can be used in a wide range of applications including diagnosis benchmark generation and bioinformatics simulation. More concrete examples of the topology generation process for diagnosis are demonstrated in Section VII and VIII.

When using an explanatory model, we first restrict the possible algorithms based on *Model Focus* (cf. column 2 of Table I), i.e., whether the domain D provides information to to generate a model from topological properties, or using an optimization approach given the system's global objective function. We briefly discuss these two types of approaches.

1) Topology-Based Generators: Given the wide range of graph generators defined in the literature, e.g., [15], [58], we have selected four of the most important approaches, i.e., the small-world graph (SWG), Preferential Attachment (PA), Spatial Preferential Attachment (SPA) and Partial Duplication (PD) models. Actually, these models show the general and fundamental principles underlying topologies of real-world systems, and we can extend them and achieve higher fidelity by introducing richer sets of domain-specific external parameters. Each approach has particular properties, which lend themselves to modeling particular domains with differing fidelity. We now summarize each model in turn.

SWG Model: This model aims to capture "small-world" properties observed in many real-world systems like electronic circuits [16] and power grids [19], such as low characteristic path length $\bar{L}$ relative to that of a classic random-graph (ER) model L_r ($\bar{L} \simeq \bar{L}_r$), and high clustering coefficient $\bar{C}$ relative to that of a ER model $\bar{C}_r$ ($\bar{C} \gg \bar{C}_r$). The SWG generator extends a regular ring lattice with a set of random connections determined by a rewiring probability p_r. We adopt the extended SWG approach of [21], which can model the arbitrary mean degrees $\bar{k}$ that occur in real systems, and not just integral mean degrees, as in the standard generator. $p_r \simeq 0$ corresponds to a regular graph, and $p_r \simeq 1$ corresponds to a random graph; graphs with real-world structure occur in between these extremes. Figure 3 depicts the graph generation process, where we control the proportion of random edges using a rewiring probability p_r. By continually increasing p_r, the regularity and modularity of the generated graph will keep decreasing, more long-range links and nodes with higher degree will appear, and consequentially characteristic path length $\bar{L}$ will become smaller.

PA Model: This model focuses on capturing the power-law of the degree distribution, using an WWW-inspired generation process [18]. Starting with n_0 isolated nodes, at each $t = 1, 2, ..., n - n_0$ a new node v_j with m_{new} links is added to the graph. The probability $P(v_i, v_j)$ that a link will connect v_j to an existing node v_i is linearly proportional to the actual degree d_i of the node v_i.

TABLE I

TOPOLOGY GENERATION APPROACHES. INPUT PARAMETERS FOR GENERATION ALGORITHMS ARE AS FOLLOWS: n—NODE NUMBER; m—EDGE NUMBER; p_r—REWIRING PROBABILITY; α—SPATIAL FACTOR; g_s—SEED NETWORK; p_d—DUPLICATION PROBABILITY; λ_i—TRADE-OFF WEIGHT; d—SUBGRAPH SIZE

Model Class	**Model Focus**	**Generation Algorithm**	**Key Properties**	**Parameters**	**Recommended Applications**	**Computational Cost**
Explanatory	Topological Properties	Small-world Graph (SWG)	Exponential degree distribution	n, m, p_r	Technological systems	Low
		Preferential Attachment (PA)	Power law degree distribution	n, m	WWW, social and citation networks	Low
		Spatial Preferential Attachment (SPA)	Power law degree distribution with cutoff	n, m, α	Spatial technological systems	Medium
		Partial Duplication (PD)	Power law degree distribution	n, m, g_s, p_d	Biological systems	Low
	Functional Optimization	Multi-constraint Optimization (OPT)	Power law degree distribution with cutoff	λ_i	Technological and transportation systems	High
Descriptive	Topological properties	dK-series	All degree correlations in d-sized subgraphs	d	Technological and biological systems	High

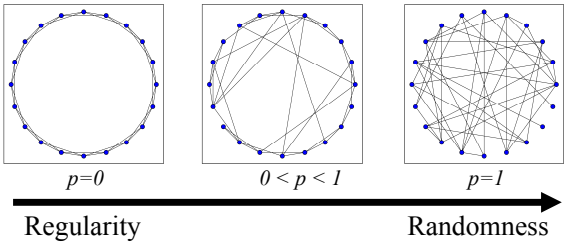

Fig. 3. Generating a small-world graph from a regular ring lattice with rewiring probability p_r.

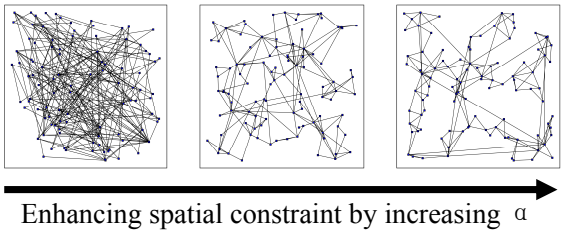

Fig. 4. Graphs generated by the SPA model by enhancing the spatial constraint.

SPA Model: The SPA model extends the PA model with a parameter α that improves the ability to capture networks with spatial constraints embedded in physical space, such as electronic circuits, telecommunications networks, and transportation networks [15]. In the SPA model, the node position is chosen randomly in a 2-D square space with uniform density. Connections of a new node v_j with each existing node v_i are established with probability $P(v_i, v_j) \propto d_i w_{ij}^{-\alpha}$, where w_{ij} is the spatial (Euclidean or Manhattan) distance between the node positions, d_i is the degree of the node v_i, and $\alpha \geq 0$ is tunable parameter used to adjust spatial constraint and shape the connection probability in the preferential attachment process. When $\alpha = 0$ the model corresponds to the standard PA model. By continually increasing α, the modularity of the generated graph will keep increasing, and fewer long-range links and high-degree nodes will appear, and consequentially characteristic path length $\bar{L}$ will become larger. Finally the degree distribution will degrade from the power-law distribution to the exponential distribution with sharper cutoff. Figure 4 displays the SPA graph generation by adjusting the geometric constraint. Similarly, we also extend the preferential attachment process in order to match the mean degree $\bar{k}$ of the real circuit.

PD Model: The PD model aims to capture the duplication mechanism which is a dominant evolutionary force in shaping biological networks [58], in contrast to other mechanisms such as preferential attachment. Given a initial seed network G_s, the network is updated by randomly choosing a node v_i, adding a duplicate of v_i, called node v_j, and connecting v_j to each neighbor of v_i with probability p_d. This model and its variants have been widely applied on bioinformatics applications related to protein interaction networks (PINs) and TRNs.

2) Optimization-Based Generators: Rather than explicitly replicate of statistical properties, the *Optimization approach* (OPT) use an optimization framework to model the mechanisms driving network growth or evolution. The OPT model formulates a weighted objective function over conflicting system properties ξ_i and weights λ_i, e.g., $f = \sum_{i=1}^{n} \xi_i \cdot \lambda_i$, and trades off the properties using the weights λ_i. For example, in some systems embedded in physical space, such as electronic circuits and brain networks, the topological structures are shaped and optimized under two conflicting constraints: information transmission steps (characteristic path length $\bar{L}$) and cost of constructing connection (average wire length $\bar{W}$) [51], [48], [62]. The objective function is formulated as follows: $f = \lambda \bar{L} + (1 - \lambda)\bar{W}$, where $0 \leq \lambda \leq 1$. For this OPT model, the optimization function concentrates only on minimizing the total wire length at $\lambda = 0$, and a regular network emerges with a nearly uniform degree and high characteristic path length $\bar{L}$. At $\lambda = 1$, the optimization function concentrates only on minimizing the average shortest path length, and a star-like network emerges with highly connected hubs. The topology graphs with the power law distributions with cutoffs should emerge when $0 < \lambda < 1$ [49], [50]. The optimization process is looking for a solution that minimizes the above objective function at an appropriate value of λ. This approach

can also give rise to power-laws in graph degree distributions with cutoffs under appropriate values of λ [49], [50]. Starting from a connected random network in which nodes are evenly put on a 2-D square, we use simulated annealing to search for the minimum cost of the objective function [63], [64]. In each annealing rearrangement step, an edge is randomly selected and rewired. In rearrangements, duplicated edges and self-loops are not allowed, to ensure that no node will be disconnected or isolated.

B. Topology Generation using Descriptive Models

The *dK-series Model* generator [45] has as its primary input parameter an integer d, which allows one to specify all degree correlations within d-size subgraphs of a given graph G[6]. $1K$ captures the degree distribution P_k and is equal to the generalized random graph (GRG) approach [15], [59]. $2K$ randomly generates synthetic graphs by maintaining of the joint degree distribution of the given topology graph G, and $3K$ considers interconnectivity among triples of nodes.

Generally, the set of $(d+1)K$-graphs is a subset of dK-graphs, and larger values of d further constrain the number of possible graphs. Given a descriptive dK-series algorithm, we generate a synthetic model $\hat{G}$ by increasing the input parameter d until the generated graph $\hat{G}$ matches the properties of the real-world graph G with sufficient fidelity in terms of the specified metrics Φ. Increasing values of d capture progressively more properties of G, at the cost of more complex representation of the probability distribution and dramatically increasing computational complexity.

[45] found that the $d = 2$ case is sufficient for most practical purposes, while $d = 3$ essentially reconstructs the Internet AS- and router-level topologies exactly in terms of regular graph metrics. Our experiments show similar results on the TRN of the E.coli and electronic circuits [60].

Another dimension in model selection encompasses trade-offs between: (1) complexity of a model and the number of metrics it tries to reproduce, and (2) its explanatory power and associated generality. Although the dK-series model generally can capture regular topological metrics better than explanatory models due to the number of constraints imposed, we cannot use it to discover laws governing the topology growth process of a particular system. It lacks predictive and rescaling power necessary for benchmark generation. Our experiments on diagnosis model generation also showed that the dK-series model is not flexible enough for fitting more complicated join-tree metrics.

VI. System Functional Model Generation

To generate a functional model, we assign components to G, and then merge the functional equations for each assigned system component.

[6]Actually, a large number parameters are needed for every value of d in real implementations, but d is the governing parameter.

A. 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 and outgoing arcs for the node. Hence, given a node $v \in G$ with i inputs and o outputs, we assign a component, denoted $\Delta_Z(i, o, \tau, \mathcal{B}_Z, w)$ where τ denotes the type (e.g., AND-gate, OR-gate), $\mathcal{B}_Z$ defines the functionality (behavioral equations) of component Z, and w the weights assigned to variables, e.g., probabilities assigned to the component failure modes of Z.

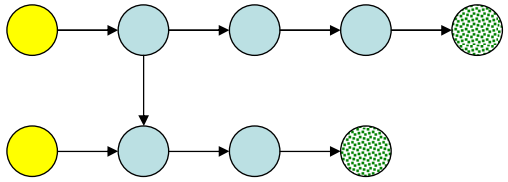

Fig. 5. Graph created for simple circuit topology. Inputs are denoted by yellow nodes, components by blue nodes, and outputs by dotted green nodes.

Example 1. Consider that the topology generation process has created a graph G, as shown in Figure 5. This graph has been created with appropriate proportions of input, component and output nodes, based on our domain analysis. Given this structure as input, together with a component library and component statistics, we can now assign components to G.

Given a node $v \in G$, we randomly assign to v a suitable component with probability based on the computed component distribution. 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).

B. 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 $O_{X,1} = O_{Y,2}$. Once this is done, we merge the component functional descriptions to generate the system functional model $\mathcal{B}$. At present, we assume that we can simply take the union of the component functional descriptions; in future work we plan to explore systems for which functional composition is more complicated.

VII. Case Study 1: ISCAS-Benchmark Circuits

This section summarizes 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 optimization [8]. 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.

A. Domain Analysis

1) Component Analysis: The ISCAS-85 benchmark circuits are presented in netlists of fundamental logic gates, which provide a standard, non-hierarchical representation specifying both network topology and functionality (in terms of the functionality of primitive gates) [65]. Our component analysis revealed that only seven types of primitive gates (NAND, NOR, NOT, AND, OR, XOR and BUFF) appear in ISCAS-85 circuits, and in general each circuit contains only four or five types of gates. Figure 6 shows several of the gates that we use in our component library, together with the gates' functionality (in terms of truth-tables).

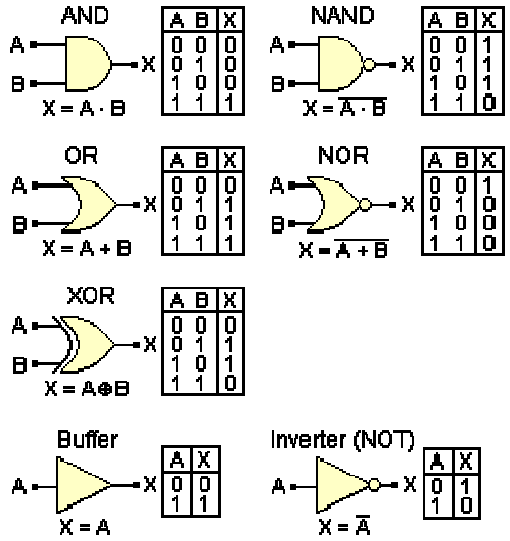

Fig. 6. Partial component library for combinatorial digital circuit domain. Each gate also has an associated truth-table defining the gate's functionality.

The NAND gates are most common components in every ISCAS-85 circuit. For instance, the % of NAND gates in C17, C432 AND C1355 is 100, 74 and 84%, respectively. One possible reason for the prevalence of the NAND gate is that it is the cheapest gate to manufacture. Additionally, NAND gates alone can be used to reproduce the functions of all the other logic gates. Figure 7 shows how an XOR functional block can be implemented by four NAND gates, and the prevalence in C1355 of many NAND gates may be due to the XOR functions it repeatedly performs.

Another property of note is that the same type of component may have various numbers of inputs. For example, in C432 most AND gates have two inputs, but a small number of AND gates have four, eight and even nine inputs. We need to carefully consider above circuit component statistics in system functional model generation.

2) Topological Analysis: Cancho et al. found *small-world graph* patterns for a small collection of electronic circuits, and observed the power law tails with cutoffs in degree distributions [16]. Figure 8 shows cumulative degree distributions for the full suite of ISCAS-85 benchmark circuits in log-log scale. We can see that most circuits exhibit long tails with cutoffs in their degree distributions. Existing analysis has conjectured that the cutoffs in power law degree distributions

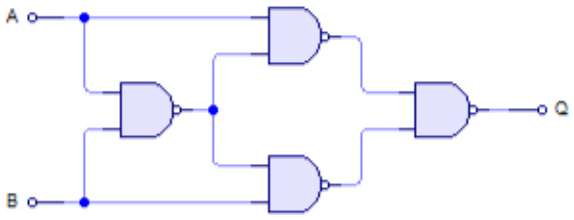

Fig. 7. A 2-input XOR functional block implemented by 4 NAND gates.

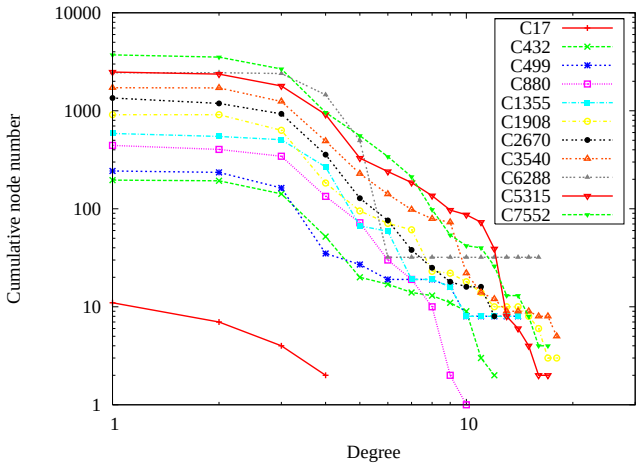

Fig. 8. Cumulative Degree distributions of ISCAS-85 benchmark circuits in log-log scale

might result from the presence of spatial constraints limiting the number of links when connections are costly [20], and this has been confirmed by further studies on diverse networks such as the Internet, power grids, transport networks and brain networks [20], [16], [66], [67], [68], [48].

In circuit design, wire length has been treated as the prime parameter for performance evaluation since it has a direct impact on several important design parameters [51], [69]. Recent research on circuit placement showed that the wire length of real circuits exhibits a power law distributions [51], [47], [62]. Another driving force underlying circuit design is timing. Many design cost metrics can be treated as technological parameters that can be optimized by trading off delay and wire length [62]. The delay of signal transmission among components can be approximately simplified as the characteristic path length.

In an electronic circuit, a cluster of components corresponds to components that together serve a particular task, e.g., a subsystem; the relatively small number of connections between clusters corresponds to the fact that sub-systems are typically loosely-coupled. As shown in Table II, the characteristic path length of each ISCAS85 circuit is close to that of the $1K$ random graph with corresponding size. Figures 9 and 10 show different views of a typical ISCAS-85 benchmark circuit, C432. It is important to note the density of shortcut edges joining nodes that have long paths (based on the circle connectivity). This circular network view displays such connections clearly, and demonstrates that the overall graph distance or

TABLE II
THE CHARACTERISTIC PATH LENGTHS OF REAL CIRCUITS AND CORRESPONDING RANDOM GRAPHS.

Circuit	real	random
C432	4.5309	4.3333
C499	4.6475	4.4053
C880	6.9756	5.4689

characteristic path length for this network will be relatively low due to such connections.

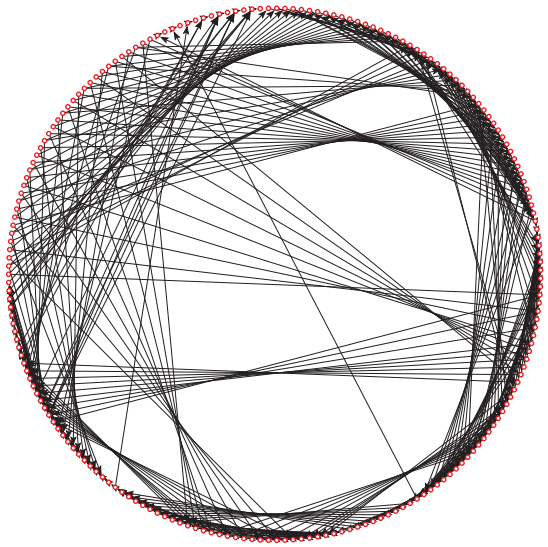

Fig. 9. The directed graph depicting the topology of circuit C432, displayed in a circular view.

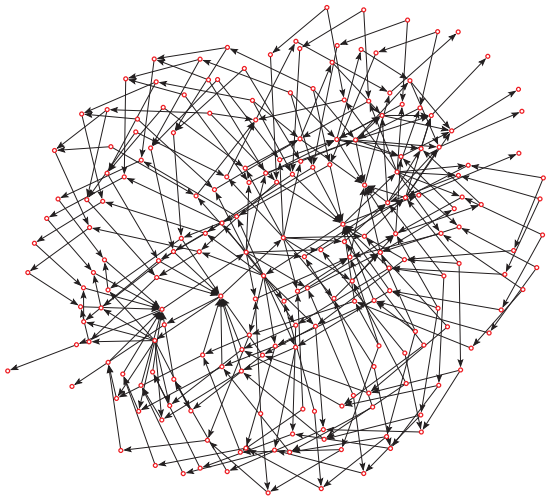

Fig. 10. The directed graph depicting the topology of circuit C432, displayed in a regular view.

3) Objective Metric Selection: This section addresses ways in which we analyse the fidelity of the synthetic models. The primary metric that we use is derived from the MBD inference task.

MBD Auto-Generation Task: Given a real-world model $(\mathcal{B}, G)$, the objective of MBD auto-generation is to create an "equivalent" synthetic model $(\hat{\mathcal{B}}, \hat{G})$ that minimizes $|\gamma_A(\hat{\mathcal{B}}, OBS) - \gamma_A(\mathcal{B}, OBS)|$, where A is an MBD inference algorithm that has complexity $\gamma_A(\mathcal{B}, OBS)$ when computing a probability-minimal diagnosis given $\mathcal{B}$ and observations OBS.[7]

As noted earlier, the treewidth of the topology graph is the key parameter for determining inference complexity. The treewidth is closely related to the largest clique size $\mu(\mathcal{T})$ of G [32], which is the parameter we adopt as a complexity measure for an MBD model defined in terms of propositional logic or as a BN.

B. Topology Generation

1) Explanatory Model Approach: We generated topology graphs using the steps shown below.

Step 1: Based on the evidence of power-laws with cutoffs in degree distribution and wire length, the SPA model combining preferential attachment with the constraint of spatial layout is a plausible candidate for topology generation of circuits. The OPT model can give rise to power laws with cutoffs in both degree distribution and wire length distribution under appropriate λ. We, along with Barthelemy [70] have found that, under appropriate parameters, the SPA model can generate structures similar to that of the OPT model. However, the computational cost of model-generation using the OPT model is significantly higher than that of using the SPA model, so we use the SPA model as an efficient alternative of the OPT model in practical applications. The SWG model is also possible choice to fit the *small-world graph* pattern observed in circuits. The SWG model naturally has sharp cutoff in its exponential degree distribution, and can vary the tail length of degree distribution in a limited range.

Step 2: we automatically optimized parameters in each model to match the $\mu(\mathcal{T})$ of real circuits. Experiments showed that two selected models can both match real circuits with appropriate parameters. For example, the typical circuit C432 can be matched by the SWG model with $p_r \simeq 0.28$ [21] as shown in Figure 12, the SPA model with $\alpha \simeq 3.7$ as shown in Figure 13. Figures 14, 15, 16 and 17 show the results of some other circuits.

Step 3: since both the SPA and SWG model fit the real circuits well in terms of $\mu(\mathcal{T})$, we can further refine the model selection by other topological metrics, such as degree distribution P_k. As shown in Figure 18, the SPA model can match real circuits better than can the SWG model in terms of P_k.

2) dK-series Approach: As shown in Table IV, when $d = 3$ the dK-series model can match almost all common circuit topological metrics perfectly, as also occurs in the case of the TRN and Internet modeling [45].

Example 2. *Given a desired system with n components, we generate the required topology, as shown in Figure 11(a). The*

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

topology of Figure 11(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.

C. Functional Model Generation

We illustrate how we construct a circuit functional model using a continuation of our running example. We show how we create functional models using propositional logic and BNs.

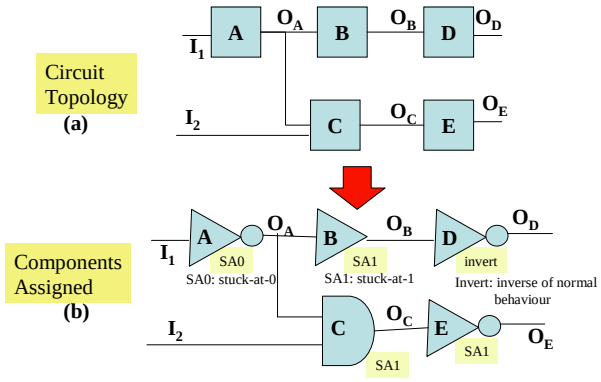

Fig. 11. Schematic of simple electronic circuit.

Example 3. *Propositional logic:* Given the topology of the 9-node graph G of Figure 5, we assign components to G as follows:

- for each input-node we assign an input;
- for each output-node we assign an output;
- for each component-node we assign a component based on the distribution over component types.

Figure 11(a) depicts the schematic of the circuit with two inputs, I_1 and I_2, with the output of component i denoted by O_i, and arbitrary components A, B, C, D and E. Given the component distribution, Figure 11(b) shows the circuit with instantiated components.

Finally we generate the system functionality in terms of the union of the component functions. For our diagnostic application, we generate the functional description as the union of the components' normal-mode equations (and potentially failure-mode equations). In the following, we define the normal-mode equations for the components Inverter, Buffer and AND:

Inverter i:	$(M_i = OK) \wedge (I_i \Rightarrow \neg O_i)$;
Buffer i:	$(M_i = OK) \wedge (I_i \Rightarrow O_i)$;
AND i:	$(M_i = OK) \wedge [(I_{i1} \wedge I_{i2}) \Rightarrow O_i]$
	$(M_i = OK) \wedge [(\neg I_{i1} \vee \neg I_{i2}) \Rightarrow \neg O_i]$

If we rename appropriate inputs and outputs, then we obtain the composed equations:

$(M_A = OK) \wedge (I_1 \Rightarrow \neg O_A)$; $(M_B = OK) \wedge (O_A \Rightarrow O_B)$;
$(M_D = OK) \wedge (I_B \Rightarrow \neg O_D)$; $(M_E = OK) \wedge (I_C \Rightarrow \neg O_E)$;
$(M_C = OK) \wedge [(O_A \wedge I_2) \Rightarrow O_C]$;
$(M_C = OK) \wedge [(O_A \wedge I_2) \Rightarrow \neg O_C]$.

Example 4. *Bayesian network:* In a Bayesian Network (BN) [71], 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.

Given the component assignment of Figure 11(b), we assign a distribution to failure-mode values by assuming that normal behavior is highly-likely, i.e., $Pr\{C_i = OK\} \simeq 0.99$, and faulty behavior is unlikely, i.e., $Pr\{C_i \neq OK\} \simeq 0.01$. Figure 11(b) depicts the instantiated failure-mode for the components in shaded boxes: Components B, C and E have SA1 fault-modes[8], 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. The distributions required are: $Pr(O_A|I_1, M_A)$, $Pr(O_B|O_A, M_B)$, $Pr(O_C|I_2, O_A, M_C)$, $Pr(O_D|O_B, M_D)$, and $Pr(O_E|O_C, M_E)$. Table III shows a sample distribution for $Pr(O_A|I_1, M_A)$.

TABLE III
TRUTH-TABLE FOR INVERTER A, WITH SA0 FAULT-MODE (WHERE THE OUTPUT IS f INDEPENDENT OF THE INPUT).

M_A	I_1	Pr(O_A)=(t,f)
OK	t	(.05,.95)
OK	f	(.75,.25)
sa0	t	(.05,.95)
sa0	f	(.15,.85)

D. Analysis of Synthetic Model

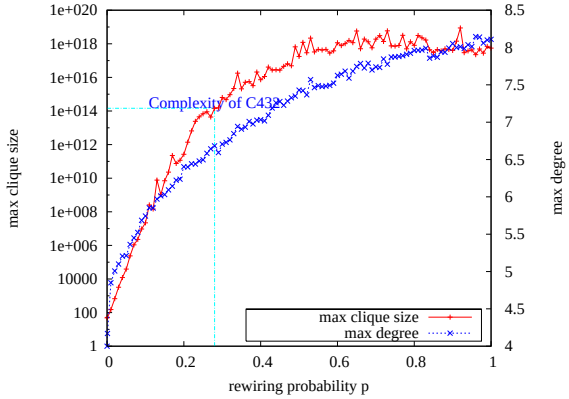

Fig. 12. The inference complexity and maximal degree of the SWG model corresponding to the circuit C432 (averaged over 100 runs).

1) Explanatory Model Approach: As shown in Figures 12, 13, 14, 15, 16 and 17, the tail lengths of the degree distributions are highly correlated with inference complexity, and the tail of P_k must be modeled well, since it defines the high-degree nodes that contribute to large cliques in the join-tree, and hence high complexities using join-tree metrics. The PA and PD model generate degree distributions with tails that are longer than those of real circuits, and therefore the resulting models have inference complexity higher than those of the corresponding real circuits.

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

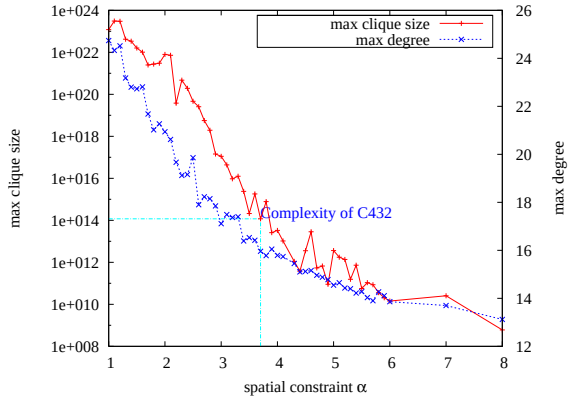

Fig. 13. The inference complexity and maximal degree of the SPA model corresponding to the circuit C432 (averaged over 100 runs).

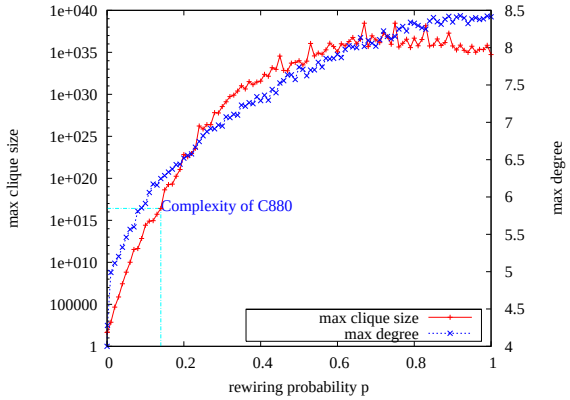

Fig. 16. The inference complexity and maximal degree of the SWG model corresponding to the circuit C880 (averaged over 100 runs).

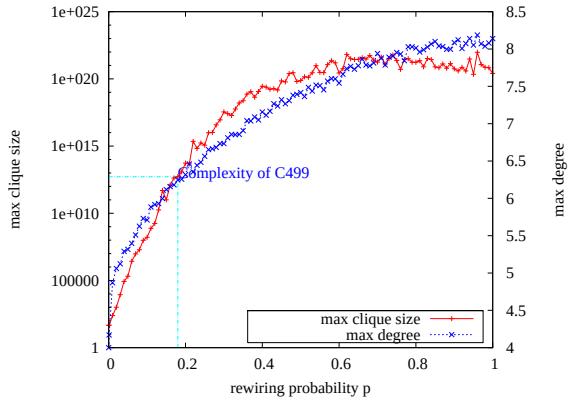

Fig. 14. The inference complexity and maximal degree of the SWG model corresponding to the circuit C499 (averaged over 100 runs).

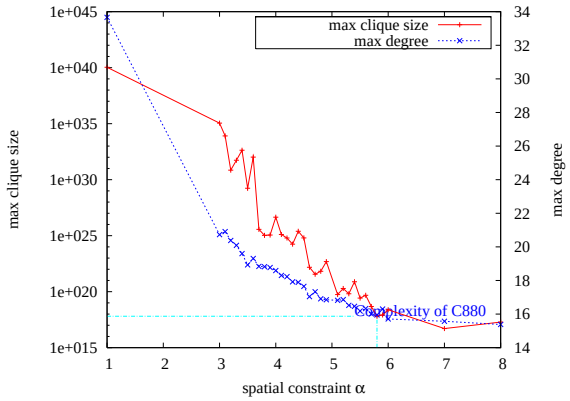

Fig. 17. The inference complexity and maximal degree of the SPA model corresponding to the circuit C880 (averaged over 100 runs).

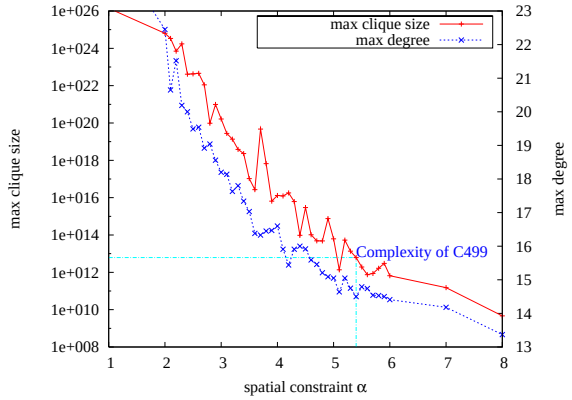

Fig. 15. The inference complexity and maximal degree of the SPA model corresponding to the circuit C499 (averaged over 100 runs).

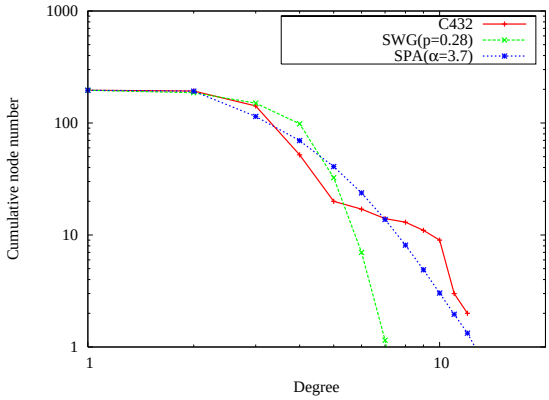

Fig. 18. Cumulative degree distributions of C432 and graphs generated by the SWG ($p = 0.28$) and SPA ($\alpha = 3.7$) model (averaged over 100 graphs).

TABLE IV
THE STATISTICS ON THE AVERAGE CLUSTERING COEFFICIENT $\bar{C}$, CHARACTERISTIC PATH LENGTH $\bar{L}$, AND s-METRIC OF CIRCUITS AND THE GRAPHS GENERATED BY THE CORRESPONDING 3K-SERIES MODEL (AVERAGED OVER 100 GRAPHS).

Model	L	C	s-Metric
C432	4.5309	0.003118	6896
3K	4.4008	0.003118	6896

TABLE V
THE INFERENCE COMPLEXITY OF C432 AND CORRESPONDING dK-SERIES MODELS ($d = 1, 2, 3$). ALL VALUES OF THREE MODELS ARE AVERAGED OVER 100 GRAPHS RESPECTIVELY.

Model	C432	1K	2K	3K
max clique size	1.4e14	8.9e17	3.3e16	1.8e16

2) dK-series Approach: Although this model captures the degree distribution and other regular topological properties well, Table V shows, however, that $d = 3$ provides insufficient fidelity to match $\mu(\mathcal{T})$ metrics for MBD benchmark generation; it also shows that increasing d can generate random graphs with increasing levels of fidelity of inference complexity. For $d > 3$ the computational complexity increases dramatically, and the size of the generated random-graph ensemble decreases exponentially as well. In this case, the dK-series model is unsuitable for diagnosis benchmark generation, compared with the SPA and the SWG model.

3) Diagnosis Complexity Analysis: Table VI lists the inference complexity data of the ISCAS-85 benchmark circuits. For all circuits of non-trivial size, the maximum clique size is very large, indicating that the diagnostics inference complexity for multiple-fault diagnoses would be high.

VIII. CASE STUDY 2: PROCESS-CONTROL SYSTEMS

This section summarizes experimental results comparing the structure and diagnostic inference complexity properties of auto-generated models with a real pulp mill benchmark model developed by Castro and Doyle [6], which consists of modular representations of unit operations in a complete pulp mill. The pulping process benchmark is based on a nonlinear dynamic mathematical model of an actual pulp mill process. The benchmark can used for several as a basis for studying several process-system tasks, including modeling, control, estimation and fault diagnosis [6].

TABLE VI
THE MBD INFERENCE COMPLEXITY IN TERMS OF THE LARGEST CLIQUE SIZE IN THE COMPILED BN MODEL.

circuit	node number	edge number	max clique size
C17	11	12	16
C432	196	336	1.40737488355328E14
C499	243	408	8.796093022208E12
C880	443	729	1.15292150460684698E18
C1355	587	1064	7.0368744177664E13
C1908	913	1497	4.6116860184273879E18
C2670	1350	2075	2.4758800785707605E27
C3540	1719	2936	3.56811923176489 97E44
C5315	2485	4386	6.73998666678766E66
C6288	2448	4800	2.037035976334486E90
C7552	3718	6144	2.9230032746618058E48

TABLE VII
THE STATISTICS OF MAJOR COMPONENTS IN THE PULP MILL BENCHMARK.

Component	Statistics
Valve	60%
Condenser	10%
Evaporator	6%
Tower	4%
Tank	4%
Washer	3%
Filter	2%
Causticizer	2%

A. Domain Analysis

1) Component Analysis: The primary goal of a pulp mill is to produce pulp of a given Kappa number, which specifies the "brightness" of a pulp stream, while minimizing energy costs, utilities and chemical make-up of each stream. Pulp mills can be divided in two major areas: fiber line and chemical recovery. In a typical pulp mill, the major units of operation are: a digester, pulp washers, oxygen tower, storage vessels, bleaching towers, evaporators, recovery boiler, smelt dissolving tank, clarifiers, slaker, causticizers and lime kiln [6].

According to the detailed schematic diagrams in [6], there are 130 basic physical components and about 180 connections in the pulp mill benchmark, and the most common basic components are valves, which are used to connect components in and between various key units. Table VII lists the statistics of the major components used in the pulp mill benchmark.

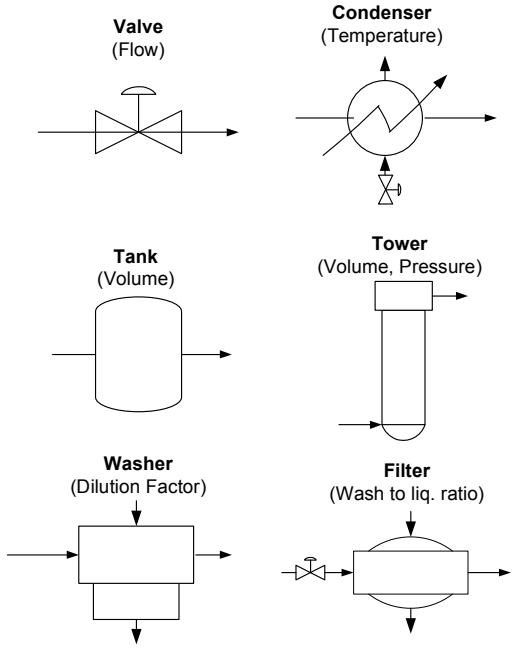

Fig. 19. Partial component library for pulp mill process control domain. Each component has specific process measurement variables associated with the component's functionality, as shown under the component's name.

Figure 19 shows several typical devices used in our pulp mill process control component library, together with the variables representing the devices' primary functional role.

Only the pulp washers are modeled by algebraic equations. All other units are modeled by ordinary differential equations (ODE) or partial differential equations (PDE) [6].

2) Topological Analysis: Based on the flowsheet and detailed schematic diagrams in [6], we decomposed the pulp mill benchmark into fundamental device components, and generated the underlying topology graph according to the physical connections among these fundamental components. As shown in Figure 29, the topology graph of the pulp mill displays a clear power law degree distribution with cutoff, which is discovered in many technological complex systems. Figures 20 and 21 show different views of the topology graph of the pulp mill, and both figures show that most connections are local connections between near neighbors, and there are few long range connections. The pattern displayed in both figures are consistent with the modular structure of the topology of the pulp mill. The detailed schematic diagrams in [6] show that the pulp mill consists of 8 loosely-coupled submodules, in each of which nodes are densely connected. We also found that the characteristic path length $\bar{L}$ of the topology graph is much larger than that of corresponding random graph.

In our topological analysis, we only consider connections between physical components; however, some components, such as valves and condensers, have an embedded control interface in addition to the inlet and outlet flow. The topology shown in Figures 20 and 21 ignores the embedded control interfaces of these components. If we treat each embedded control interface as an "input" component, we can generate a new topology graph with 232 nodes and nearly 300 edges. Figures 22 and 23 show different views of the corresponding topology graph, which share a similar pattern with the topology graph in Figures 20 and 21. As shown in Figure 30, the new topology graph displays a clear power law degree distribution with cutoff as well.

If we compare the two circular views of the pulp mill (Figure 20 and 22) with that of a typical circuit, C432 (Figure 9), we see that the pulp mill has far fewer shortcut edges than does the circuit. As noted earlier, these shortcut edges connect components that would otherwise be connected by long paths. Hence, these figures clearly show that the pulp mill has a longer mean graph distance or characteristic path length than does the circuit. Also, even with the control structure, the essential linear aspect of the process-control system is the dominant structural feature.

3) Objective metric selection: The main objective of our study was to provide as close a comparison of the system models of the circuit and pulp mill domains as possible. To achieve this in terms of diagnostic complexity, we can create propositional logic and Bayesian network (BN) diagnostic models for the system-level pulp mill, since this was the class of model we generated for the circuits. Although this class of model differs from the FDI approach taken for most diagnostics studies of the pulp mill, e.g., [72], [73], system-level BN models can provide useful diagnostics for the global behaviors of a pulp mill.

To build a qualitative model for the pulp mill benchmark, we discretized all of our variables. For example, the input signals (manipulated variables) are scaled and constrained between ± 1.0, and output signals are scaled based on the nominal/steady-state value of outputs, and the maximum range of change of the outputs [6]. The scaled continuous-valued variables can be further converted into discrete-valued variables according to the specification of the pulp process.

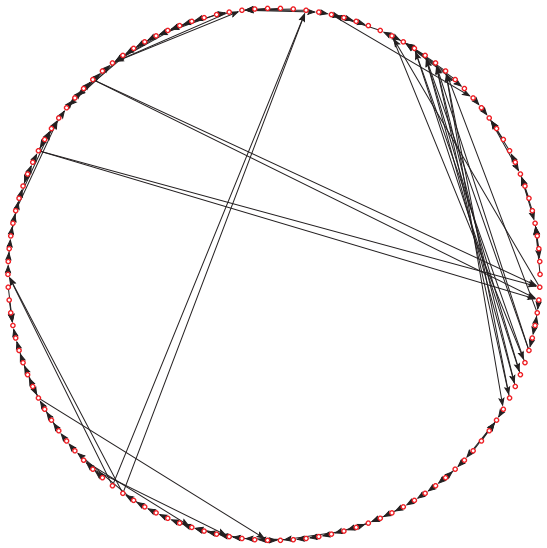

Fig. 20. The topology graph of the physical components in the pulp mill displayed in a circular view.

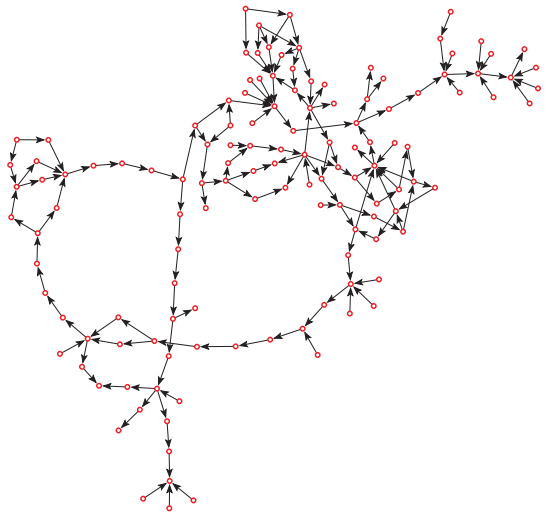

Fig. 21. The topology graph of the physical components in the pulp mill displayed in a regular view.

In using the logic and BN approaches for our experiments on the pulp mill, we selected $\mu(\mathcal{T})$ as the objective metric, as done for our circuit experiments.

B. Topology Generation

1) Explanatory Model Approach: We generated topology graphs using the steps shown below.

Step 1: Since the pulp mill displays a power law degree distribution with cutoff, the SPA model is a natural candidate for topology generation. According to the connection pattern

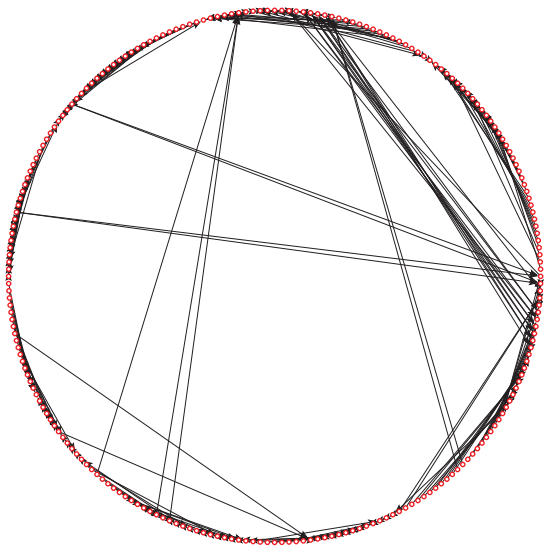

Fig. 22. The topology graph of the physical components and embedded control interfaces in the pulp mill displayed in a circular view.

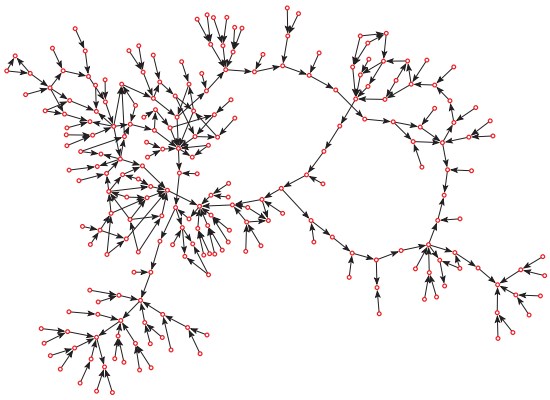

Fig. 23. The topology graph of the physical components and embedded control interfaces in the pulp mill displayed in a regular view.

shown in Figure 20 and Figure 22, the SWG model seems another possible candidate. Due to the large characteristic path length $\bar{L}$, the p_r of the SWG model should have rewiring probability with a relatively small value.

Step 2: We automatically optimized parameters in each model to match the $\mu(\mathcal{T})$ of the above two types of topology graphs of the pulp mill benchmark respectively.

The topology graph of physical components: Figure 25 shows that the pulp mill benchmark can be matched by the SWG model with $p_r \simeq 0.15$. But as shown in Figure 26, the corresponding SPA model cannot match the low $\mu(\mathcal{T})$ of the pulp mill benchmark. By increasing α, the SPA model generally can produce the graphs with sharper cutoffs in degree distributions and consequent lower $\mu(\mathcal{T})$, but as shown in Figure 26, tuning α becomes counterproductive when $\alpha > 5$ due to the limited size of the pulp mill benchmark. As shown in Figure 29, the SPA model can match the pulp mill benchmark much better than can the SWG model in terms of P_k. The data in Figure 29 are averaged over 100 graph instances, and demonstrate that the SPA model generally can closely match the degree distribution of the pulp mill benchmark, although a small fraction of graph instances, with overly long tails in their degree distributions, contribute to a high $\mu(\mathcal{T})$.

The topology graph of physical components and control interfaces: The results are similar to those of the topology graph of physical components. Figure 27 shows that the pulp mill benchmark can be matched by the SWG model with $p_r \simeq 0.08$. As shown in Figure 28, the SPA model cannot match the low $\mu(\mathcal{T})$ of the pulp mill benchmark. Figure 30 shows similar results on P_k as well.

2) dK-series Approach: The dK-series model is not flexible enough for diagnosis benchmark generation, due to the number of constraints imposed, so we only analyze the approach in terms of the regular topological metrics. When $d = 3$, the dK-series model cannot capture the basic topological properties of the pulp mill benchmark, although the same approach can perfectly match all regular topological metrics of the Internet, TRNs and electronic circuits. The characteristic path lengths $\bar{L}$ of the both topology graphs of the pulp mill benchmark are around 30% larger than that of the corresponding $3K$ models.

C. Functional Model Generation

According to the process described in the previous section, we can generate a synthetic topology graph G. Figure 24(a) shows a snippet of the generated topology graph G (including both physical components and control interfaces) with nodes A, B, C and D. We associate with each node in G a component, based on the number of inputs and outputs for the node. All four nodes in Figure 24(a) have two inputs and one output, and we randomly select a component for each node with probability which is proportional to its frequency in the actual pulp mill. As shown in Figure 24(b), the four nodes are randomly associated with the valve and condenser respectively, which are two types of the most common components in the pulp mill as shown in Table VII.

In the component library of the pulp mill domain, a component is denoted $\Delta_Z(i, o, \tau, \mathcal{B}_Z, w_Z)$ where τ denotes the type (e.g., valve, condenser, tank), $\mathcal{B}_Z$ defines the functionality (behavioral equations) of component Z, and w_Z the probabilities assigned to the component failure modes of Z. For example, the valve component V is represented as $SD_V(2, 1, valve, \mathcal{B}_V, w_V)$ in the component library.

According to the technological specification of the valve V, we can discretize the flow rate of the V based on appropriate thresholds, and define the flow rate as following three states: f (flow), rf (reduced flow) and nf (no flow) [74]. The possible fault modes M_V for valves can be modeled as: OK, sc(stuck closed), lk (leaking), and hb (half-blocked) [74]. We assume the control condition $ctrl_V$ of the valve V (i.e., whether it has been commanded to be open or closed) is known. The propositional equations for a functional model $\mathcal{B}_V$ of the valve V can be defined as shown in Table VIII. The group of formulas in Table VIII describing the behavior of a valve V shows that the output flow out_V depends, beside on the

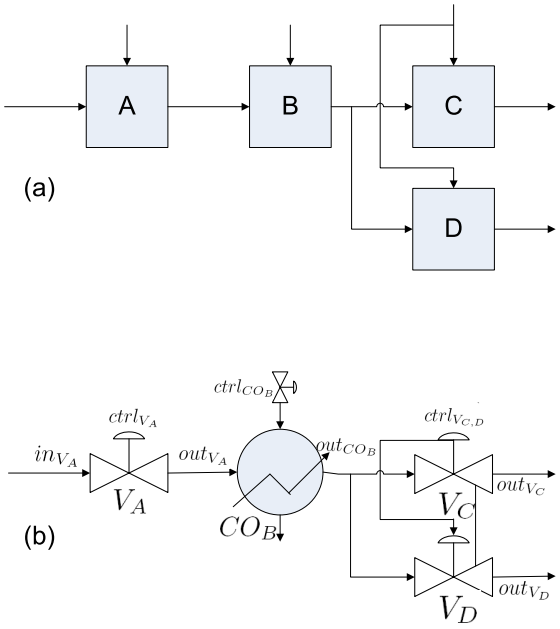

Fig. 24. Schematic of a snippet of the synthetic pulp mill benchmark.

TABLE VIII
THE BEHAVIORAL EQUATIONS $\mathcal{F}_V$ FOR THE VALVE COMPONENT V.

$[M_v = OK] \wedge [in_V = *] \wedge [ctrl_V = o] \Rightarrow [out_V = *]$
$[M_v = OK] \wedge [in_V = *] \wedge [ctrl_V = c] \Rightarrow [out_V = nf]$
$[M_V = so] \wedge [in_V = *] \wedge [ctrl_V = o] \Rightarrow [out_V = *]$
$[M_V = so] \wedge [in_V = *] \wedge [ctrl_V = c] \Rightarrow [out_V = *]$
$[M_V = sc] \wedge [in_V = *] \wedge [ctrl_V = o] \Rightarrow [out_V = nf]$
$[M_V = sc] \wedge [in_V = *] \wedge [ctrl_V = c] \Rightarrow [out_V = nf]$
$[M_V = lk] \wedge [in_V = f] \wedge [ctrl_V = o] \Rightarrow [out_V = rf]$
$[M_V = lk] \wedge [in_V = rf] \wedge [ctrl_V = o] \Rightarrow [out_V = rf]$
$[M_V = lk] \wedge [in_V = nf] \wedge [ctrl_V = o] \Rightarrow [out_V = nf]$
$[M_V = lk] \wedge [in_V = *] \wedge [ctrl_V = c] \Rightarrow [out_V = nf]$
$[M_V = hb] \wedge [in_V = f] \wedge [ctrl_V = o] \Rightarrow [out_V = rf]$
$[M_V = hb] \wedge [in_V = rf] \wedge [ctrl_V = o] \Rightarrow [out_V = rf]$
$[M_V = hb] \wedge [in_V = nf] \wedge [ctrl_V = o] \Rightarrow [out_V = nf]$
$[M_V = hb] \wedge [in_V = f] \wedge [ctrl_V = c] \Rightarrow [out_V = rf]$
$[M_V = hb] \wedge [in_V = rf] \wedge [ctrl_V = c] \Rightarrow [out_V = rf]$
$[M_V = hb] \wedge [in_V = nf] \wedge [ctrl_V = c] \Rightarrow [out_V = nf]$

behavioral mode M_V, on the input flow in_V and on the current control condition $ctrl_V$ of the valve V; for example, the first formula of the group asserts that, if valve V is in the OK behavioral mode and the control condition is o(open), it will simply report its input flow in_v (for example, f) to its output. On the contrary, if valve V is in the hb behavioral mode (half-blocked), its control condition is c (closed) and the input in_V is f , the output out_V is rf (reduced flow). In general, we use the symbol $*$ to denote any admissible value for a variable to limit the number of formulas [74].

Similarly, we can also define the functional model for condensers, on which the temperatures of the flow are measured.

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. The system model for our model fragment consists of three sets of valve equations (as depicted in Table VIII), together with a set of condenser equations.

To construct a BN model, we need conditional probability distributions for the components V_A, CO_B, V_C, V_D, rather than propositional equations. The set of distributions needed for our model fragment is:

$$Pr(out_{V_A}|in_{V_A}, ctrl_{V_A}, M_{V_A})$$
$$Pr(out_{CO_B}|out_{V_A}, ctrl_{CO_B}, M_{CO_B})$$
$$Pr(out_{V_C}|out_{CO_B}, ctrl_{V_{C,D}}, M_{V_C})$$
$$Pr(out_{V_D}|in_{CO_B}, ctrl_{V_{C,D}}, M_{V_D}).$$

For the distributions, we assume that all the faulty behavioral modes are equally likely, i.e., $Pr\{M_V \neq OK\} = 0.01$, and have a much smaller probability than the nominal mode,i.e., $Pr\{M_V = OK\} = 0.99$. We extract the remaining probabilities based on analysis of the differential equation models.

D. Analysis of Synthetic Model

1) Explanatory Model Approach: Although the SWG generally cannot match the degree distribution well, it is a good and flexible model for fitting small-scale systems in terms of $\mu(\mathcal{T})$. Even for the systems without "small-world" properties like the pulp mill benchmark (with large characteristic path length) the SWG model with a small p_r is appropriate. In contrast, the SPA model is not suitable for generating diagnosis benchmark of the pulp mill, which has relatively low complexity.

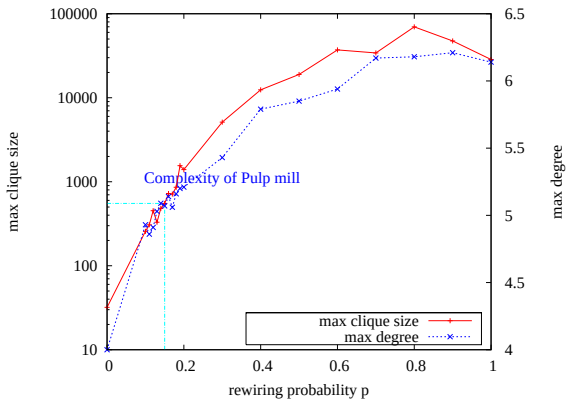

Fig. 25. The average-case inference complexity and average maximal degree of the SWG model fitting the Pulp Mill (averaged over 100 runs)

2) dK-series Approach: In existing topological analysis of complex systems [60], [45], researchers found $k = 3$ is generally sufficient for reproducing regular topological properties perfectly. But we found that the characteristic path length $\bar{L}$ of the pulp mill benchmark is much larger than the corresponding $3K$ model, and the result shows that the underlying topology of the pulp mill benchmark has some organizational principles different from the Internet, TRNs and electronic circuits. To distinguish underlying mechanisms will be a interesting topic in the future.

3) Diagnosis Complexity: The topology graph shown in Figures 20 and 21, and the topology graph shown in Figures 22 and 23 of the pulp mill have the same $\mu(\mathcal{T})$ value 512, a really

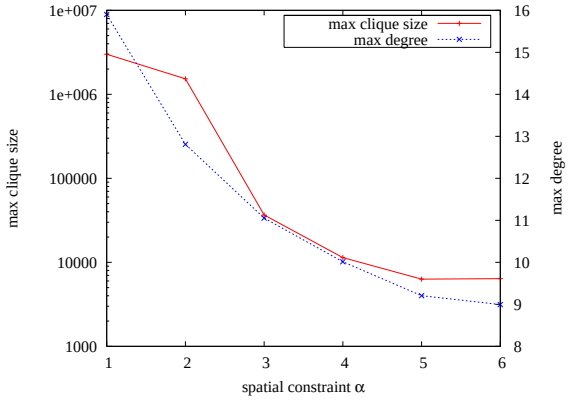

Fig. 26. The average-case inference complexity and average maximal degree of the SPA model fitting the Pulp Mill (averaged over 100 runs)

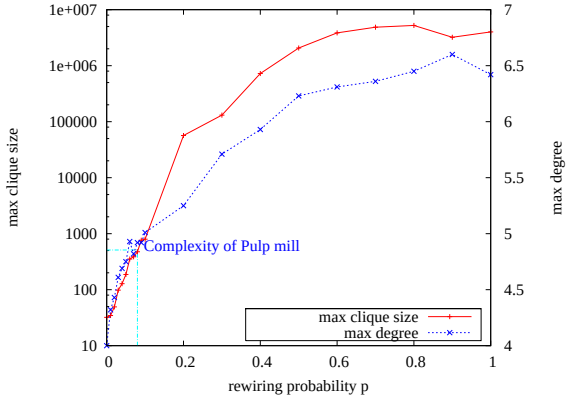

Fig. 27. The average-case inference complexity and average maximal degree of the SWG model fitting the Pulp Mill (averaged over 100 runs)

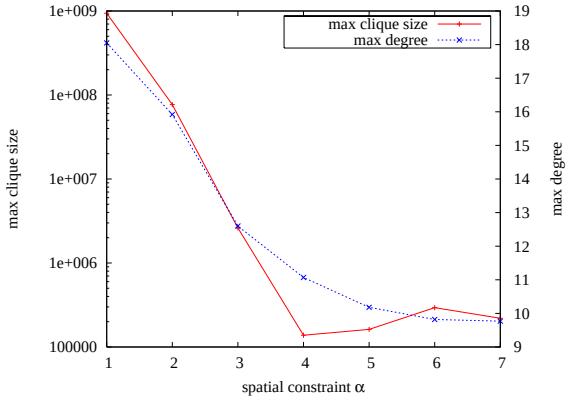

Fig. 28. The average-case inference complexity and average maximal degree of the SPA model fitting the Pulp Mill (averaged over 100 runs)

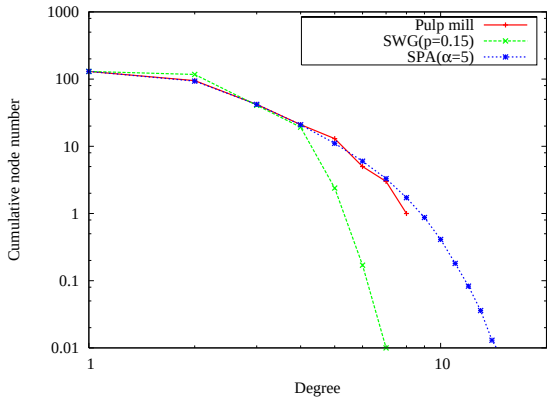

Fig. 29. Cumulative degree distributions of the pulp mill and graphs generated by the SWG ($p = 0.15$) and SPA ($\alpha = 5$) model (averaged over 100 graphs).

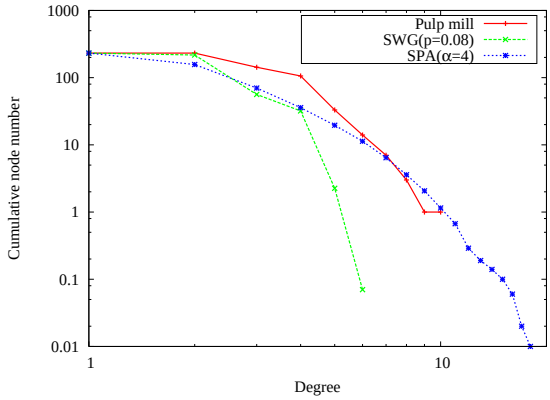

Fig. 30. Cumulative degree distributions of the pulp mill and graphs generated by the SWG ($p = 0.08$) and SPA ($\alpha = 4$) model (averaged over 100 graphs).

small value compared with that of C432 (1.4 e14), although C423 has approximately the same number of components as the pulp mill benchmark.

IX. Related Work

A. Automated Model Generation

The topology-generation method we adopt was originally developed based on the theory of random graphs and complex networks–see [15], [17] 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.

This work is most closely related to domain-specific model generators, which exist for circuits [51] and biological interaction kinetics models [7]. Our approach is different from either of these approaches, in that we make no prior assumptions about domain properties, but rather compute the domain properties necessary for model generation. Our model generation approach differs from related work in VLSI auto-generation, e.g., [51], in several ways. The VLSI approach emphasizes circuit design for circuit optimization and simulation after

placement and routing; in contrast, our approach focuses on topological and organizational principles of circuits, and can be used for a wider variety of applications, including the diagnostics applications we report.

This paper improves upon the model generation approach of [21] in several ways. First, it explicitly defines a domain-analysis phase. Second, it extends and improves upon the topology generation algorithms for creating the underlying system structure, and examines a wider range of metrics for empirically evaluating synthetic networks.

B. Model Composability and Modeling Tools

Compositional modelling uses a set of functional component models, together with a specification of component interactions (called a "scenario" in [10]) to generate useful (mathematical) models. Our approach differs from that of [10] in that we create the system structure, or scenario using model generators instead of manual work. 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 socio-economic, ecological and biological systems [7], [75], [76], [38].

Biswas *et al.* explore the use of bond graphs [77], [78] for compositional modeling: [31] describes how a meta-programmable visual modeling tool, called the Generic Modeling Environment (GME), can be used for compositional modeling, and in [79] they describe the use of this approach for system simulation. Along similar lines, Bouamama *et al.* [80] describe a tool for designing FDI algorithms for thermofluid processes, together with a library of component models [81]. Our proposed model-generation approach is fully compatible with bond-graph modeling and model generation for FDI.

Several authors have addressed the formal aspects of compositional modeling. For example, Gossler *et al.* [82], [25] address formal modeling for compositional modeling from the point of view of heterogeneous interaction and execution, and extend the framework in [30]. Denckla and Mosterman [29] have defined a formal block-diagram language. They have applied it to hybrid systems [29]; in addition, [83] provides a translational semantics for block diagrams using the programming language Haskell. Rather than focus of the formal aspects of compositional modeling, we have addressed issues of model generation, assuming such formal properties will hold.

A variety of *causal (or block-oriented) languages* have been developed, and have been implemented in tools such as Modelica [11] and Simulink [13]. Such languages often support object-oriented tools such as Dymola [84] and Modelica [11], [85]. Our proposed model-generation approach can be viewed as an extension of such modeling tools, in that our proposed tool can act as a generator for models using the component libraries associated with such tools. The objectives of our approach are different than those of such tools: we start with existing models to generate additional benchmark models; most modeling tools create new models from scratch.

X. Conclusion

A. Summary

This article has described a tool, CoSyMGen, for generating functional models that have real-world topology. CoSyMGen provides a key advance for benchmark generation: given a set of exemplar system models and a component library, it can automatically analyze the exemplar models to generate synthetic benchmark models, of arbitrary size, with properties that closely match those of the exemplar models. In addition, users can provide a range of experimental metrics for the synthetic models, such as diagnostics complexity, or simulation fidelity, in order to fine-tune the synthetic models. This method circumvents the problems with using random graphs for experimental studies of diagnostics, and provides an alternative to manually developing suites of benchmark models.

We have demonstrated CoSyMGen on two real-world domains, those of diagnostics for discrete circuits and pulp mill process-control. Our experiments have shown that CoSyMGen can be used for two domains with significant differences, from the discrete-valued models typically studied in the MBD community, to the continuous-valued process-control models typically studied in the FDI community. We argue that 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.

B. Discussion

In addition to the benchmark generation capabilities, the domain analysis approach of CoSyMGen reveals several useful system properties that can be used to guide diagnosticians in the selection of inference and analysis techniques. In this sense, one can view domain analysis as the *meta-analysis of domain properties*. For example, in comparing the circuit and process-control domains, our topological analysis has revealed significant differences in topology, which affect the choice of system-level inference algorithms which use structure-based approaches, e.g., methods based on SAT or CSP techniques [86]. Circuits have large clusters and high treewidth, leading to intractability for any algorithm whose complexity is governed by treewidth. Hence, one will need to use additional techniques, such as component abstraction [87], or stochastic algorithms, e.g., [88], to render inference more tractable in this domain. In contrast, process-control systems have relatively small clusters and low treewidth, leading to tractability for any algorithm whose complexity is governed by treewidth.

In future work, we plan to extend our domain analysis to cover properties governing continuous-valued inference parameters, analogous to the structure-based parameters (cluster-size and treewidth) that our current system covers. In addition, we plan to make this tool available as an extension to compositional modeling tools for benchmark generation from existing component libraries.

References

[1] H. Kautz, B. Selman, and J. Hoffmann, "SatPlan: Planning as Satisfiability," *5th International Planning Competition*, 2006.

[2] P. D. Kundarewich and J. Rose, "Synthetic circuit generation using clustering and iteration," in *FPGA '03: Proceedings of the 2003 ACM/SIGDA eleventh international symposium on Field programmable gate arrays*. New York, NY, USA: ACM Press, 2003, pp. 245–245.

[3] D. Stroobandt, P. Verplaetse, and J. van Campenhout, "Towards synthetic benchmark circuits for evaluating timing-driven CAD tools," in *ISPD '99: Proceedings of the 1999 international symposium on Physical design*. New York, NY, USA: ACM Press, 1999, pp. 60–66.

[4] M. Bhushan and R. Rengaswamy, "Comprehensive design of a sensor network for chemical plants based on various diagnosability and reliability criteria. 1. framework," *Industrial and Engineering Chemistry Research*, vol. 41, no. 7, pp. 1826–1839, 2002.

[5] D. Dvorak and B. Kuipers, "Process monitoring and diagnosis: a model-based approach," *Expert, IEEE [see also IEEE Intelligent Systems and Their Applications]*, vol. 6, no. 3, pp. 67–74, 1991.

[6] J. J. Castro and F. J. D. III, "A pulp mill benchmark problem for control: Problem description," *J. Proc. Cont.*, vol. 14, pp. 17–29, 2004.

[7] 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 of structure learning algorithms." *BMC Bioinformatics*, vol. 7, 2006.

[8] J. E. Harlow, "Overview of Popular Benchmark Sets," *IEEE Design and Test of Computers*, vol. 17, no. 3, pp. 15–17, 2000.

[9] M. Bartyś, R. Patton, M. Syfert, S. de las Heras, and J. Quevedo, "Introduction to the DAMADICS actuator FDI benchmark study," *Control Engineering Practice*, vol. 14, no. 6, pp. 577–596, 2006.

[10] J. Keppens and Q. Shen, "On compositional modelling," *Knowl. Eng. Rev.*, vol. 16, no. 2, pp. 157–200, 2001.

[11] J. Broenink, "Bond-graph modeling in Modelica," *ESS97-European Simulation Symposium*, 1997.

[12] ——, "20-sim Software For Hierarchical Bond-Graph/Block-Diagram Models," *Simulation Practice and Theory*, vol. 7, no. 5-6, pp. 481–492, 1999.

[13] *MATLAB Simulink*, www.mathworks.com/products/simulink/.

[14] N. Mazzocca and V. Vittorini, "Compositional Modeling of Complex Systems: Contact Center Scenarios in OsMoSys," *Applications And Theory Of Petri Nets 2004: 25th International Conference, ICATPN 2004, Bologna, Italy, June 21-25, 2004: Proceedings*, 2004.

[15] S. Boccaletti, V. Latora, Y. Moreno, M. Chavez, and D.-U. Hwang, "Complex networks : Structure and dynamics," *Physics Reports*, vol. 424, no. 4-5, pp. 175–308, 2006.

[16] R. F. i. Cancho, C. Janssen, and R. V. Solé, "Topology of technology graphs: Small world patterns in electronic circuits," *Physical Review E*, vol. 64, no. 4, p. 046119, Sep 2001.

[17] M. E. J. Newman, "The structure and function of complex networks," *SIAM Review*, vol. 45, p. 167, 2003.

[18] A. L. Barabasi and R. Albert, "Emergence of scaling in random networks," *Science*, vol. 286, pp. 509–512, 1999.

[19] D. J. Watts and S. H. Strogatz, "Collective dynamics of 'small-world' networks." *Nature*, vol. 393, no. 6684, pp. 440–442, June 1998.

[20] L. A. Amaral, A. Scala, M. Barthelemy, and H. E. Stanley, "Classes of small-world networks." *Proc. Natl. Acad. Sci. USA*, vol. 97, no. 21, pp. 11 149–11 152, October 2000.

[21] G. M. Provan and J. Wang, "Automated benchmark model generators for model-based diagnostic inference," in *IJCAI*, 2007, pp. 513–518.

[22] R. Reiter, "A Theory of Diagnosis from First Principles," *Artificial Intelligence*, vol. 32, pp. 57–96, 1987.

[23] P. Frank, "Fault diagnosis in dynamic systems using analytical and knowledge-based redundancy- A survey and some new results," *Automatica*, vol. 26, no. 3, pp. 459–474, 1990.

[24] P. Frank, S. Ding, and T. Marcu, "Model-based fault diagnosis in technical processes," *Transactions of the Institute of Measurement & Control*, vol. 22, no. 1, p. 57, 2000.

[25] G. Gößler and J. Sifakis, "Composition for component-based modeling," *Sci. Comput. Program.*, vol. 55, no. 1-3, pp. 161–183, 2005.

[26] J. de Kleer, A. Mackworth, and R. Reiter, "Characterizing diagnoses and systems," *Artificial Intelligence*, vol. 56, no. 2-3, pp. 197–222, 1992.

[27] S. Srinivas, "A probabilistic approach to hierarchical model-based diagnosis," *Proc. UAI-94*, p. 538545, 1994.

[28] P. Lucas, "Bayesian model-based diagnosis," *International Journal of Approximate Reasoning*, vol. 27, no. 2, pp. 99–119, 2001.

[29] B. Denckla and P. Mosterman, "Formalizing Causal Block Diagrams for Modeling a Class of Hybrid Dynamic Systems," *Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC'05. 44th IEEE Conference on*, pp. 4193–4198, 2005.

[30] G. Gößler, S. Graf, M. E. Majster-Cederbaum, M. Martens, and J. Sifakis, "An approach to modelling and verification of component based systems," in *SOFSEM (1)*, 2007, pp. 295–308.

[31] E. Manders, G. Biswas, N. Mahadevan, and G. Karsai, "Component-oriented modeling of hybrid dynamic systems using the Generic Modeling Environment," *Proc of the 4th Workshop on Model-Based Development of Computer Based Systems, Potsdam, Germany*, 2006.

[32] A. Darwiche, "Model-based diagnosis using structured system descriptions," *J. Artif. Intell. Res. (JAIR)*, vol. 8, pp. 165–222, 1998.

[33] J. Keppens and Q. Shen, "Causality Enabled Compositional Modelling of Bayesian Networks," *Proceedings of the 18th International Workshop on Qualitative Reasoning about Physical Systems*, pp. 33–40, 2004.

[34] A. Breunese, J. Broenink, J. Top, and J. Akkermans, "Libraries of Reusable Models: Theory and Application," *Simulation*, vol. 71, no. 1, p. 7, 1998.

[35] E. Posse, J. de Lara, and H. Vangheluwe, "Processing causal block diagrams with graph-grammars in AToM 3," *European Joint Conference on Theory and Practice of Software (ETAPS), Workshop on Applied Graph Transformation (AGT)*, pp. 23–34, 2002.

[36] L. Pradeep and K. Khosla, "Object-Oriented Libraries of Physical Components in Simulation and Design," *Simulation*, vol. 10, p. 11.

[37] F. Cellier, "Hierarchical non-linear bond graphs: a unified methodology for modeling complex physical systems," *Simulation*, vol. 58, no. 4, p. 230, 1992.

[38] P. Mendes, W. Sha, and K. Ye, "Artificial gene networks for objective comparison of analysis algorithms." *Bioinformatics*, vol. 19 Suppl 2, October 2003.

[39] C. T. Harbison, B. D. Gordon, T. I. Lee, N. J. Rinaldi, K. D. Macisaac, T. W. Danford, N. M. Hannett, J.-B. Tagne, D. B. Reynolds, J. Yoo, E. G. Jennings, J. Zeitlinger, D. K. Pokholok, M. Kellis, A. P. Rolfe, K. T. Takusagawa, E. S. Lander, D. K. Gifford, E. Fraenkel, and R. A. Young, "Transcriptional regulatory code of a eukaryotic genome," *Nature*, vol. 431, no. 7004, pp. 99–104, 2004.

[40] S. S. Shen-Orr, R. Milo, S. Mangan, and U. Alon, "Network motifs in the transcriptional regulation network of escherichia coli," *Nat Genet*, vol. 31, no. 1, pp. 64–68, May 2002.

[41] D. E. Zak, G. E. Gonye, J. S. Schwaber, and F. J. Doyle, "Importance of input perturbations and stochastic gene expression in the reverse engineering of genetic regulatory networks: insights from an identifiability analysis of an in silico network." *Genome Res*, vol. 13, no. 11, pp. 2396–2405, November 2003.

[42] N. Barkai and S. Leibler, "Circadian clocks limited by noise," *Nature*, vol. 403, no. 6636, pp. 267–8, Jan. 2000.

[43] R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon, "Network motifs: simple building blocks of complex networks." *Science*, vol. 298, no. 5594, pp. 824–827, October 2002.

[44] L. Costa, F. Rodrigues, G. Travieso, and P. Boas, "Characterization of complex networks: A survey of measurements," *Advances in Physics*, vol. 56, no. 1, pp. 167–242, 2007.

[45] P. Mahadevan, D. V. Krioukov, K. R. Fall, and A. Vahdat, "Systematic topology analysis and generation using degree correlations." in *SIGCOMM*, 2006, pp. 135–146.

[46] D. Chakrabarti and C. Faloutsos, "Graph mining: Laws, generators, and algorithms," *ACM Comput. Surv.*, vol. 38, no. 1, 2006.

[47] D. Christie, P. Stroobandt, "The interpretation and application of Rent's rule," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 8, pp. 639–648, 12 2000.

[48] Y. He, Z. J. J. Chen, and A. C. C. Evans, "Small-World Anatomical Networks in the Human Brain Revealed by Cortical Thickness from MRI," *Cereb Cortex*, January 2007.

[49] R. M. D'Souza, C. Borgs, J. T. Chayes, N. Berger, and R. D. Kleinberg, "Emergence of tempered preferential attachment from optimization," *Proc. Natl. Acad. Sci. USA*, vol. 104, no. 15, pp. 6112–6117, April 2007.

[50] N. Mathias and V. Gopal, "Small worlds: How and why," *Physical Review E*, vol. 63, p. 021117, 2001.

[51] D. Stroobandt, "Analytical methods for a priori wire length estimates in computer systems," *Ph.D. dissertation: Ghent University*, 2001.

[52] N. Przulj, "Biological network comparison using graphlet degree distribution," *Bioinformatics*, vol. 23, no. 2, pp. e177–e183, 2007.

[53] L. Li, J. C. Doyle, and W. Willinger, "Towards a Theory of Scale-Free Graphs: Definition, properties, and Implications," *Internet Mathematics*, vol. 2(4), pp. 431–523, March 2006.

[54] N. Przulj, D. G. Corneil, and I. Jurisica, "Modeling interactome: scale-free or geometric?" *Bioinformatics*, vol. 20, no. 18, pp. 3508–3515, 2004.

[55] M. Middendorf, E. Ziv, and C. H. Wiggins, "Inferring network mechanisms: the drosophila melanogaster protein interaction network." *Proc. Natl. Acad. Sci. USA*, vol. 102, no. 9, pp. 3192–7, Mar 1 2005.

[56] M. Grohe and J. Flum, *Parameterized Complexity Theory*. Springer, 2006.

[57] D. Krioukov, K. Claffy, M. Fomenkov, F. Chung, A. Vespignani, and W. Willinger, "The workshop on internet topology (wit) report," *SIGCOMM Comput. Commun. Rev.*, vol. 37, no. 1, pp. 69–73, 2007.

[58] F. R. K. Chung, L. Lu, T. G. Dewey, and D. J. Galas, "Duplication models for biological networks," *Journal of Computational Biology*, vol. 10, no. 5, pp. 677–687, 2003.

[59] F. Viger and M. Latapy, "Efficient and simple generation of random simple connected graphs with prescribed degree sequence." in *COCOON*, 2005, pp. 440–449.

[60] J. Wang and G. M. Provan, "Generating Application-Specific Benchmark Model for Complex Systems," in *AAAI*, 2008.

[61] F. Hormozdiari, P. Berenbrink, N. Przulj, and S. C. C. Sahinalp, "Not All Scale-Free Networks Are Born Equal: The Role of the Seed Graph in PPI Network Evolution," *PLoS Comput Biol*, vol. 3, no. 7, July 2007.

[62] J. Dambre, "Prediction of interconnect properties for digital circuit design and technology exploration," *Ph.D. dissertation: Ghent University, Faculty of Engineering*, 2003.

[63] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science, Number 4598, 13 May 1983*, vol. 220, 4598, pp. 671–680, 1983.

[64] W. H. Press, S. A. Teukolsky, W. T. Vettering, and B. P. Flannery, *Numerical Recipes in C++. The Art of Scientific Computing*, 2nd ed. Cambridge University Press, February 2002.

[65] M. C. Hansen, H. Yalcin, and J. P. Hayes, "Unveiling the ISCAS-85 Benchmarks: A Case Study in Reverse Engineering," *IEEE Des. Test*, vol. 16, no. 3, pp. 72–80, 1999.

[66] R. Guimera and L. Amaral, "Modeling the world-wide airport network," *The European Physical Journal B - Condensed Matter*, vol. 38, no. 2, pp. 381–385, March 2004.

[67] R. Guimera, S. Mossa, A. Turtschi, and L. A. N. Amaral, "The worldwide air transportation network: Anomalous centrality, community structure, and cities' global roles," *Proc. Natl. Acad. Sci. USA*, vol. 102, p. 7794, 2005.

[68] S.-H. Yook, H. Jeong, and A.-L. Barabasi, "Modeling the internet's large-scale topology," *Proc. Natl. Acad. Sci. USA*, vol. 99, p. 13382, 2002.

[69] D. Stroobandt, "A priori wire length distribution models with multiterminal nets," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 11, no. 1, pp. 35–43, 2003.

[70] M. Barthlemy, "Crossover from scale-free to spatial networks," *Europhysics Letters*, vol. 63, pp. 915–921, 2003.

[71] F. Jensen, S. Lauritzen, and K. Olesen, "Bayesian Updating in Recursive Graphical Models by Local Computations," *Comp. Stat. Q.*, vol. 4, pp. 269–282, 1990.

[72] G. Lee, T. Tosukhowong, and J. Lee, "Fault Detection and Diagnosis of Pulp Mill Process," *Computer Aided Chemical Engineering*, vol. 21, no. B, p. 1461, 2006.

[73] A. Samantaray and S. Ghoshal, "Sensitivity bond graph approach to multiple fault isolation through parameter estimation," *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, vol. 221, no. 4, pp. 577–587, 2007.

[74] P. Torasso, "Compact diagnoses representation in diagnostic problem solving," *Computational Intelligence*, vol. 21, pp. 27–68(42), February 2005.

[75] R. Duboz and C. Cambier, "Small world properties in a dsdevs model of ecosystem," in *Proceedings of the Open International Conference on Modeling and Simulation (OICMS-2005)*, 2005, pp. 65–71.

[76] J. Keppens and Q. Shen, "Granularity And Disaggregation In Compositional Modelling With Applications To Ecological Systems," *Applied Intelligence*, vol. 25, no. 3, pp. 269–292, 2006.

[77] F. E. Cellier, H. Elmqvist, and M. Otter, "Modeling from physical principles," in *The Control Handbook*, W. S. Levine, Ed. Boca Raton, FL: CRC Press, 1995, pp. 99–108.

[78] D. Karnopp, D. Margolis, and R. Rosenberg, *Systems Dynamics: A Unified Approach*, 2nd ed. John Wiley and Sons, 1990.

[79] M. Daigle, I. Roychoudhury, G. Biswas, and X. Koutsoukos, "Efficient Simulation of Component-Based Hybrid Models Represented as Hybrid Bond Graphs," *Lecture Notes In Computer Science*, vol. 4416, p. 680, 2007.

[80] B. Bouamama, A. Samantaray, K. Medjaher, M. Staroswiecki, and G. Dauphin-Tanguy, "Model builder using functional and bond graph tools for FDI design," *Control Engineering Practice*, vol. 13, no. 7, pp. 875–891, 2005.

[81] B. Bouamama, "Bond Graph Approach As Analysis Tool In Thermofluid Model Library Conception," *Journal of the Franklin Institute*, vol. 340, no. 1, pp. 1–23, 2003.

[82] G. Gossler and J. Sifakis, "Composition for Component-Based Modeling," *Formal Methods for Components and Objects: First International Symposium, FMCO 2002, Leiden, The Netherlands, November 5-8, 2002: Revised Lectures*, 2003.

[83] B. Denckla and P. Mosterman, "Block Diagrams as a Syntactic Extension to Haskell," Tech. Rep., 2007.

[84] F. Cellier and R. McBride, "Object-Oriented Modeling of Complex Physical Systems Using the Dymola Bond-Graph Library," *Proc. ICBGM03, International Conference of Bond Graph Modeling and Simulation*, pp. 19–23, 2003.

[85] P. Fritzson, *Principles of Object-oriented Modeling and Simulation with Modelica 2.1*. Wiley-IEEE, 2004.

[86] G. Gottlob, R. Pichler, and F. Wei, "Bounded Treewidth as a Key to Tractability of Knowledge Representation and Reasoning," *Proc. AAAI*, pp. 250–256, 2006.

[87] S. Siddiqi and J. Huang, "Hierarchical diagnosis of multiple faults," *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 581–586, 2007.

[88] A. Feldman, G. M. Provan, and A. J. C. van Gemund, "Computing Minimal Diagnosis by Greedy Stochastic Search," in *Proc. AAAI*, 2008.