Exploring Graphical Models with Bayesian Learning and MCMC for Failure Diagnosis

Hongfei Wang, Wenjie Cai, Jianwen Li, and Kun He
School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, China

***Abstract*—Graphical models are powerful machine learning techniques for data analytics. Being capable of statistical reasoning and probabilistic inference, graphical models have the advantages of encoding prior knowledges into the learning procedure, and producing explainable models that can be understood and effectively tuned. In this work, we describe our exploration on the frontier of using graphical models for improving circuit diagnosis results. A statistical framework has been proposed for this aim, which builds Bayesian inference models using directed chain graphs, and structural learning models using undirected tree graphs. As a generative model, the framework integrates Markov chain Monte Carlo (MCMC) algorithm for sampling to evaluate the quality of diagnostic results. It exploits maximum-likelihood to estimate the underlying defect types, which can be informative towards the possible follow-up failure analysis. Five circuit examples demonstrate that the proposed framework achieves the same or better results over a state-of-the-art work. Moreover, our method also shows opportunities for dealing with missing features and locating root causes.**

I. Introduction

Manufactured integrated circuits (ICs) are usually tested to ensure the production quality meets expectations before shipping to customers. The portion of the ICs passing the tests out of the entire manufactured batch of ICs is called *yield*. Much endeavors in silicon industry can be attributed to bringing up the yield, which is directly related to profits. Yield learning therefore makes huge efforts to boost yield during early production volume ramping, and maintain the yield level after process maturity.

A critical practice to facilitate yield learning is logic diagnosis, which is a software-based analytics. Diagnosis maps a failure phenomenon to a set of possible defects or locations insides failing ICs, using the applied test patterns, the failure-log files collected from the testers, and the circuitry description. Owning to the advancement of technology nodes and engineering capability, modern circuits are becoming more and more sophisticated than ever before. As a result, more than often, fault models explaining the failures cannot be obtained deterministically. According to our own experience and our industrial partners, diagnostic tools integrate heuristics and statistical analysis to produce diagnosis reports. This gives the rise of using machine learning (ML) to build data-driven, statistical models to enhance diagnostic results [1]–[15].

One broad research area in ML is to learn graphical models [16]–[18], also known as probabilistic graphical models [19]. Although being successful in multiple applications including speech recognition, computer vision, genomics and proteomics, graphical models have rarely been used for failure diagnosis, except for an early work in [3]. This early exploration, however, only set foot in one particular subfield of graphical models. We provide more detailed comparison between [3] and our work in Section VI.

In this paper, a novel ML framework that we call **DIAGRAM** (**DIA**gnosis using **GRA**phical **M**odels) is proposed to improve circuit failure diagnosis results. To the best of our knowledge, `DIAGRAM` is the first work systematically leveraging multiple key techniques from graphical modeling for use in diagnosis. `DIAGRAM` is capable of predicting the sufficient amount of test-data volume for a quality fault diagnosis using Bayesian inference. More importantly, `DIAGRAM` can work with missing data (up to 5% missing) without surrendering significant loss in accuracy. Such capability comes from the fact that `DIAGRAM` trains generative models, which make it very applicable in real-world data analytics where industrial test data may not be perfectly complete and handy. `DIAGRAM` can also find the maximum likelihood structures among the failing population of circuits. The graph architectures thereby constructed can be potentially useful towards allocating scarce resources for very limited number of failure analysis.

II. Background

A. Overview of Machine Learning for Test and Diagnosis

During the past two decades, machine learning (ML) has been extensively studied to promote efficiency, efficacy, and intelligence in chip test and diagnosis. Generally, the data set used to learn a model can be represented as an $N \times M$ matrix (denoted by X), where each row is an input instance described by M features. We use X_i^j ($1 \leq i \leq N$, $1 \leq j \leq M$) to represent the j-th feature value of the i-th data instance.

If labels are available, which contain numerical values of particular interests, supervised learning can be used as the basic guideline for training. Usually, the labels can be specified in a $N \times 1$ vector (denoted by Y), corresponding to each instance X_i (i.e., a row $X(i,\cdot)$ from the matrix X). Once the training completes, given unlabelled data instances X^*, the learned model predicts its estimated labels $\hat{Y^*}$ based on any reasonable hypothesis from learning.

Example classification questions formulated from chip test and diagnosis applications include "is this JTAG access a legitimate one or an illegal attack [20] ?", "does the diagnostic report match the precise failure culprits [7] ?", "would the test-data volume collected so far sufficient for a quality diagnosis [1] ?", and "can we discard these tests without incurring noticeable defect escapes [21]–[23] ?". Other ML techniques have been applied to address real-world problems where labels are partially available. The work in [24] employs active

This work was supported by the National Natural Science Foundation of China (NSFC) under grant No. 61702473. Corresponding authors: Hongfei Wang (hongfeiw@alumni.cmu.edu); Kun He (brooklet60@hust.edu.cn).

learning, where the true/ false labels of defects can be acquired interactively or adaptively [24].

Training a model does not necessarily require the presence of labels, especially when they cannot be obtained easily. Unsupervised learning explores the underlying patterns and trends within X, irrespective of labels Y. As a typical unsupervised learning method, clustering has been shown to be highly effective for reasoning the spatial and temporal correlation (or causation) among wafers, dies, and test patterns [10]–[14], [25], [26].

B. A Brief on Graphical Models

Graphs are made up by nodes and edges. Graphical models are an important family of ML paradigms that are primarily used for probabilistic inference and causal reasoning. The nodes hold variable values from observation or sampling. The edges encode belief relationships among nodes. Fig. 1 shows examples of two most commonly used graph models: undirected model and directed model. Most graphical modeling involves the task of identifying a joint distribution of the variable nodes. In Fig. 1, the joint distribution is $p(\theta, X)$ for both models.

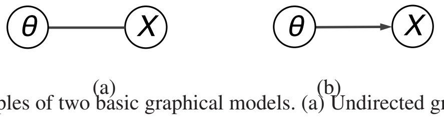

Fig. 1. Examples of two basic graphical models. (a) Undirected graph models. (b) Directed graph models.

The presence of an arrow on an edge not only differentiates between undirected and directed graphical models, but also indicates a formalization of the relationships between the linked nodes. In Fig. 1 (b), the joint distribution can be further factorized as $p(\theta, X) = p(\theta)p(X|\theta)$. For directed models, θ can be regarded as the unobserved model parameter that causes the quantities observed via X. On this occasion, we can encode any available prior knowledge into the model through the specification of $p(\theta)$. Other possible circumstances for using directed models also exist. For example, if treating both θ and X as evidences observable, the occurrence of θ precedes that of X. Likewise, chained graphical models can be configured in this sequential manner.

Graphical models are intuitive and often self-explained. This is one huge advantage comparing with many other ML techniques [16] [17] [19]. We provide a taxonomy of most graphical models in Fig. 2, though not meant to be all-inclusive.

Unlike many types of ML techniques, graphical models are seldom used for circuit test and diagnosis. One early work in [3] proposed a method to capture the root causes of failing chips, by combining the layout-aware diagnosis with Bayesian networks. Using the terminology in Fig. 2, the work [3] adopts parameter learning to build Bayesian nets with directed models for discriminative analysis, i.e., classification of the suspects against actual defects.

III. Problem Setting

When testing chips, the output responses of failing ones are recorded in case diagnosis is later performed. Insufficient

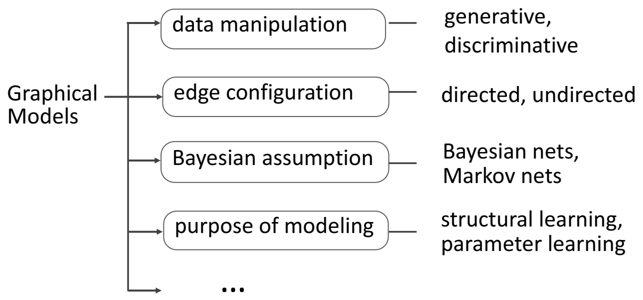

Fig. 2. Several types of graphical modeling methods.

data collection could lead to inaccurate diagnosis, while an excessive amount increases the overall cost from multiple aspects. The test measurement data volume has to be sufficient to enable accurate diagnosis. Since each output port has a fan-in logic cone, back-cone tracing is employed during diagnosis to locate possible defect sites [27]. Halting an operating ATE (automatic test equipment) at a premature stage or adopting the stop-on-first-fail test mode is likely to prevent diagnosis software from locating defect accurately. The cost of collecting test measurement data can be significant due to the extra test time incurred for going beyond the first failing pattern, the data-storage cost for high-volume products, and the expensive maintenance due to ATE wearout. Practices such as test compaction and multi-site testing, either produce the inaccurate diagnosis, or do not necessarily reduce test-data volume. Therefore, optimizing the test-data volume meets the twin goals of reduced test cost and improved diagnosis quality. It is desirable even when data-collection cost is not an issue. For software-based logic diagnosis, it is conceptually intuitive to deem that "the more data, the better result". It turns out that an excessive amount of fail data may lead to degradation of diagnostic resolution [1] [7].

The state-of-the-art solution uses a ML-based method to tackle the problem [1]. Test-data volume optimization is formulated as a binary classification task. Following the notation introduced in Section II-A, obtaining the <X,Y> is the first step to enable supervised ML. First, the test responses are processed into the "X" via the developed feature extraction technique. Altogether seven features are extracted for each of the failing test patterns of a CUT (chip under test), such as "number of test patterns that have been applied" and "total number of erroneous output bits that have been accumulated". Next, the diagnosis results are generated and subsequently processed into the "Y". A chip that has failed M_i test patterns is used for diagnosis, one failing pattern for each diagnostic result. A *golden* diagnosis result for a failing chip refers to the one generated when using all the applied tests, that is, from the first test pattern through the last failing test. An *intermediate* result refers to any outcome generated without using the entire set of test patterns. A numerical value in the label vector "Y" is produced by measuring the similarity between an intermediate diagnosis result and a golden diagnosis, using the metrics developed in [1]. Hence, Y_i^j is the label for the j^{th} failing pattern from the i^{th} CUT.

Once the data is prepared, a classification model is trained. Whenever a CUT fails a test pattern, the model is invoked to determine if the amount of test data accumulated so far is sufficient for an accurate diagnosis analysis. If yes, terminate testing; if no, continue. The termination point for each CUT is thereby dynamically determined, measured by data-volume reduction (DVR) ratio, $DVR = \frac{1}{N}\sum(1 - |t_i^T|/|T_i|) \times 100\%$, where N is the total number of chips, $|T_i|$ is the test-data volume collected for the i^{th} chip, by accumulating the number of failing output bits for all failing test patterns, from the first to the last. $|t_i^T|$ is the optimized volume calculated in a similar way, from the first pattern to the test-termination point predicted for this CUT.

IV. Graphical Models for Improving Diagnosis

A. Graphical Models for Sampling and Inference

In statistics and ML, inference generally means using the observed data with any available, prior knowledge to make reasonable predictions. A prediction can be a numerical value for an unobserved variable, or logical judgements from a decision system. DIAGRAM builds Bayesian graphical models (also called Bayesian nets) for inference.

The designed models are shown in Fig. 3. Following the problem setting and notations described in Section III, X denotes the extracted features, and Y denotes the diagnostic results. Hence, X_i^{jk} denotes the k^{th} feature extracted for the j^{th} failing pattern from the i^{th} chip. We use Z to denote the hidden defect in a failing CUT. The subscript and superscripts, i, j and k, are used for indexing CUTs, diagnostic results (from using the correspondingly failing test patterns), and features extracted from failing outputs, respectively. We will not use these indices in the following explanation unless necessary.

Fig. 3 (a) shows the basic skeleton for the graphical models. It reveals the causal relations we coded into the model: an injected defect Z leads to failing of a CUT, observed by the failing output responses, which are used to produce diagnostic results Y, and also extracted for features X. The joint distribution between X, Y, and Z can be factorized as $p(X, Y, Z) = p(Z)p(X|Z)p(Y|Z, X)$. In Fig. 3 (a), there is an arrow pointing to diagnosis Y from the features X. This is because although features X are not directly used for diagnosis, they are originally from the CUT's outputs, which produced failure-log files for diagnostic results. Hence we reserve an directed edge between X and Y. Defects in the CUT are hidden and cannot be observed directly, hence represented by filled nodes (with the gray Z's).

One key idea in graphical models with Bayesian theory is that it sees almost everything as from certain probability distributions. Static, fixed values from observation or evidence are regarded as from stochastic variables as well, only with variance approaching zero. Fig. 3 (b) represents the distribution configuration for different CUTs and diagnosis. We order the defect types into categorical numbers starting from $1, 2, ..., 5$. A specific defect injected for a CUT is randomly selected from an uniform distribution, with upper and lower bound given by θ. We set initial values as $\theta = [0, 6]$, where other reasonable

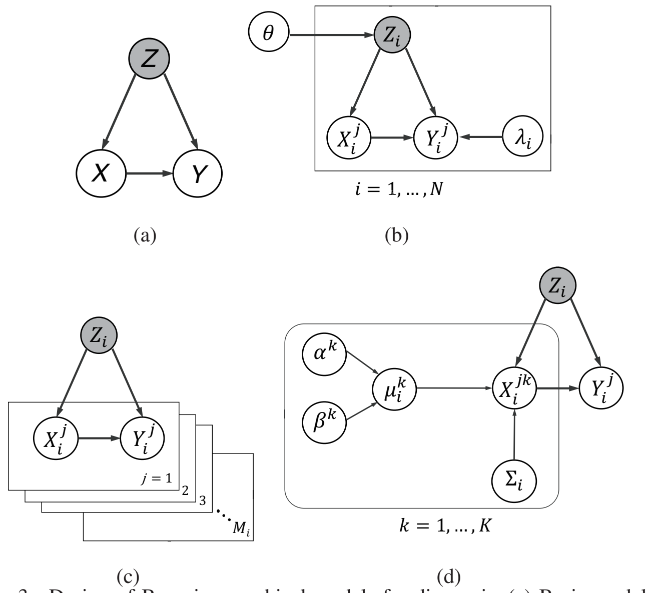

Fig. 3. Design of Bayesian graphical models for diagnosis. (a) Basic model skeleton. (b) Representation of distribution models for chips and diagnostic results. (c) Illustration of different failing test patterns, with corresponding output responses and diagnostic reports (including both intermediate and gold ones). (d) Multivariate response model representation for features extracted. The plates in (b), (c), and (d) allow a more compact representation for indexing chips, test patterns, and features extracted from failure-log data. Hidden variables are represented by filled gray nodes.

intervals can work as long as to include $[1, 5]$. A floor function is used to truncate any decimals so that the generated numbers are integers representing the defect-type category. For the i^{th} CUT, the diagnostic results are approximated by exponential distribution with input parameter λ_i. Exponential distribution has the property that the cumulative distribution function is approaching 1 with the increase of inputs. Such property roughly matches the fact that a diagnostic result Y_i^j is increasing from 0 towards 1. A typical example Y_i looks like the sequence: $0, 0, 0.1135, 0.1135, 0.5922, 0.8187, 1, ..., 1$. The increase of j's indicates that more failing patterns have been incurred, which also suggests that the current intermediate diagnostic result is becoming more similar or the same as the golden diagnosis. The initial values are $\lambda_i = 1$ for all CUTs.

The rectangular plate in Fig. 3 (b) depicts that the enclosed subgraph is repeated for a number of N chips. On the other hand, it also means that the unenclosed parameter node θ is shared by multiple nodes. The same idea and representation are also illustrated in Fig. 3 (c) and (d).

Fig. 3 (c) shows that both features X_i^j and the diagnostic result Y_i^j are updated, with the occurrence of each failing test pattern denoted by index j. $1 \leq j \leq M_i$, M_i represents the i^{th} chip failed a total number of M_i failing patterns.

Fig. 3 (d) presents the feature distribution model. The feature $X_i^{jk} (1 \leq k \leq K)$ is derived using the information starting from the first one till the the j^{th} failing test pattern, as explained in Section III. We deem a feature as observed values, comparing the hidden defect type represented by Z. Since multiple observations (i.e., the features) made on the same test (i.e., the j^{th} failing test pattern) and same CUT (i.e., the i^{th} CUT) may be correlated, we can assume that they follow a multivariate normal (MVN) distribution as,

$$X_i^j = \begin{bmatrix} X_i^{j1} \\ \vdots \\ X_i^{jK} \end{bmatrix} \sim MVN_k(\mu_i, \Sigma_i), k = 1, 2, ..., K \quad (1)$$

where the mean μ_i is a vector of size K, and the covariance matrix Σ_i is a square of size $K \times K$. The rest of the modeling task is to specify the mean vector and covariance matrix. For the k^{th} element in the μ_i vector, we construct a linear regression model as

$$\mu_i^k = \alpha^k + \beta^k Z_i \quad (2)$$

In Equation (2), the intercept α^k and slope β^k are generated from normal distributions, i.e., $\alpha^k \sim$ `Normal`(0, 10^{-4}), $\beta^k \sim$ `Normal`(1, 10^{-4}). The Σ_i matrix follows a Wishart distribution by $\Sigma_i^{-1} \sim W(R, \rho)$. The Wishart distribution serves as a vague conjugate prior for the precision matrix of multivariate normal distribution, with $\rho = K$ and $R = \rho A$, where A is a diagonal matrix of size $K \times K$. All initial values for parameters are deliberately set to be vague and least informative, as they will be iteratively updated by MCMC (explained below).

Specifying a model distributions is both an opportunity and a challenge. The model can benefit from appropriate distribution configuration, but may suffer performance degradation otherwise. Fortunately, as a generative model, `DIAGRAM` uses Markov chain Monte Carlo (MCMC) for sampling and update. Theoretical expositions on MCMC can be found in many references [16]–[19]. Here we explain the crucial ideas behind without heavy math.

The first *MC*, Markov chain, is a general approach to iteratively model the transitional distribution along the chain variables. Given sufficient iterations, a Markov chain enters into a stationary state, similar to an equilibrium system whose underlying joint distribution is little affected by the initial values. The second *MC*, Monte Carlo, is a widely applicable sampling method. By using the acceptance-rejection sampling theory, Monte Carlo especially suits the scenarios when data distribution is not easily obtained. Putting the two MCs together, MCMC methods are used to approximate the posterior distribution of model parameters or interested variable by sampling with reasonable choices. For Bayesian models, much of the inference, prediction, and estimation come from posterior distribution, which is the product of prior distribution and likelihood distribution. Intuitively, Markov chain minimizes the negative impact of any possible improper prior configurations, while Monte Carlo makes the likelihood more reasonable through sampling and estimation. Moreover, MCMC particularly fits into the diagnosis problem here. Sampling techniques can be viewed as generating virtual population of failing chips and symptoms, with similarities to the provided observation (the actually real data as X, Y, and Z). A sequence of correlated variables represented by chain models depicts the correlation among chips, tests, and diagnostic results. Variable interaction is initialized by prior configuration and finally smoothed and determined by likelihood. Therefore, `DIAGRAM` leverages MCMC as a powerful tool for inference.

B. Graphical Models for Structural Learning

Bayesian analysis is a powerful tool in data science, yet it requires that the data used obeys independent and identical distribution (i.i.d.). Validation of such assumption can be very difficult by nature, thus in many cases we just assumed so by approximation [28]. However, In circumstances like wafer excursion, process variation, change of fabrication equipments, etc., the assumption of i.i.d. may fail to be true. Insisting on using Bayesian methods puts the learned model on the fringe of a potential problem - even a statistical model barely fits the imperfect test data and occasionally works, it depicts things far from the physical, real silicon world. To put short, Bayesian models can be both efficient and effective towards determining the go/ no-go of a chip by analyzing test correlation, but may be in a state of inertia when catching the critical anomalies hidden in the power traces and heat-dissipation features.

`DIAGRAM` circumvents the i.i.d. constraint using Chow-Liu algorithm [29]. It searches for the maximum weighted spanning trees in a graph. A tree can be viewed as a directed acyclic graph. The dependence among graph nodes are thereby calculated via maximum likelihood estimation. Deriving such graph structure does not need to follow the Bayesian approach. Hence, this functionality in `DIAGRAM` does not need to meet the prerequisite of i.i.d. in order to find the dependence structure, which is found to be positively correlated with the defect types.

V. Experiments

The experiment setting is adopted from [1] and kept consistent for comparison. Five benchmark circuits are performed to demonstrate the viability of our method for a variety of circuit types. The circuits include c499 and c7552 from the ISCAS'85 benchmarks, s5378 and s7552 from the ISCAS'89 sequential circuits, and b12 from the ITC'99 suite. The defect simulation framework from [30] is used to create a population of realistic failures from various benchmark designs, which have simulation responses from layout-injected defects. Randomly-selected defects are generated for each possible defect type that includes open, bridge, cell defects, transistor stuck-open, and transistor stuck-closed. Defects are injected, one at a time, into the layout of each benchmark. An extracted netlist of each defective circuit is then simulated at the circuit-level using 100% stuck-at test sets generated by a commercial ATPG tool. The resulting simulation responses form the virtual test data for the failing population of circuits created.

`DIAGRAM` is programmed in R language with multiple packages [31] providing graphical model primitives. Part of `DIAGRAM`'s inference functionality is built upon the BUGS (Bayesian inference Using Gibbs Sampling) project [32] [33]. The datasets are divided into two disjoint sets: 90% are randomly sampled for training, the rest 10% for testing. The experiment is repeated 10 times (10-fold cross validation) to calculate averaged values. On average DIAGRAM achieves the same level of performance as the work in [1], i.e., an accuracy $> 90\%$ with a DVR (data-volume reduction) $\geq$30.2%.

DIAGRAM produces generative models, implying that the learning procedure can tolerate missing data. We randomly replace a number of extracted features in X with NA (denoting the absence of observed values) for 2% and 5%. The results on c7552 are presented in Fig. 4, comparing to the method in [1]. Plots of the other benchmarks resemble this one. The work [1] uses decision-tree method for test-termination prediction, which is shown to be the winner out of all the ML methods experimented. For decision trees, as a classic ML method, we treat missing data in two ways: one is to delete the data instances with missing values and go on training the models using the remaining data, the other is to interpolate the missing values with the means calculated from the rest and train on the combination of both real and synthetic data. The first one essentially means a model is fed with few examples during training, while the second one may derail the training procedure from the reality. Either way, we see a degradation of accuracy in Fig. 4 for all three methods. DIAGRAM's performance also suffers from missing data, since data with missing values provides fewer evidences for the observed nodes in the graphical model. However, DIAGRAM manages to maintain the accuracy $> 90\%$, which is least impacted by the missing data comparing with generic ML method in [1].

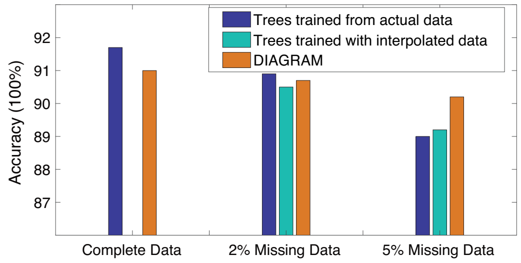

Fig. 4. Using DIAGRAM for inference with missing data, comparing with decision-tree learning. The DVR is maintained as $\geq$21.0%.

We now presents more close inspection into DIAGRAM's inference mechanisms. Fig. 5. Remember in Section IV-A, the model parameters such as alpha and beta are initialized in a vague fashion. Fig. 5 (a) shows the history plot of these parameters, each using the first element (alpha[1] and beta[1]) as examples. The traces are plotted against iteration number, showing that the designed Markov chain converges to its stationary distribution with approximately 600 iterations onwards. Most model parameters in a converged Markov chain typically look like this: a random variable scattering about a stable value [34]. The plot provides a hint on the number of iterations that should be adopted to make the chain graph stable, though empirically >2K iterations would be sufficient in our experiments.

DIAGRAM makes inference based on learned distribution, meaning that we can also infer the hidden node values as model parameters. Fig. 5 (b)$\sim$(d) reveal the defect type

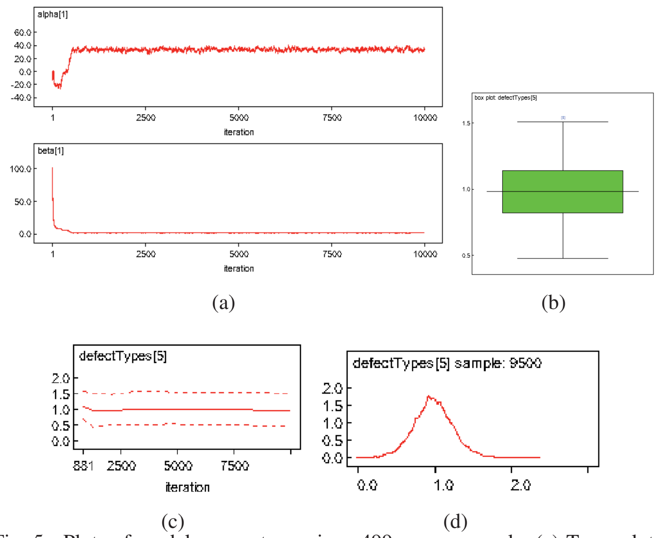

Fig. 5. Plots of model parameters using c499 as an example. (a) Trace plots of linear regression model parameters. (b) Boxplot with 2.5% and 97.5% percentiles for the mean value, (c) Running quantile against iterations, and (d) Density of an inferred node denoting defect types.

inferred for the 5th chip as a hidden, unobserved node in the graphical model illustrated in Fig. 3 (i.e.,Z_i, where $i = 5$). In Section IV-A, we encode the defect types into whole integers (1, 2, ...) as categorical factors. From Fig. 5 (b)$\sim$(d), this inferred value is "1", indicating the chip has an open defect injected before test-response simulation.

Note determining the actual defect types can be intrinsically hard. Previous works resort to physical layouts for help [3] [7] [24]. Fig. 6 reveals such difficulty by showing the results from using a variety of clustering techniques [17]. None of them can produce a homogeneous cluster, where the majority failing chips belonging to only one defect type. We also performed Radom Forests (with 100 trees) for classification, only with accuracy $< 47\%$. On the contrary, the structural learning functionality in DIAGRAM is able to identify different defect types with $> 83\%$ accuracy. DIAGRAM finds a spanning tree structure in an undirected graph, as illustrated in Fig. 7 for b12 circuit as an example.

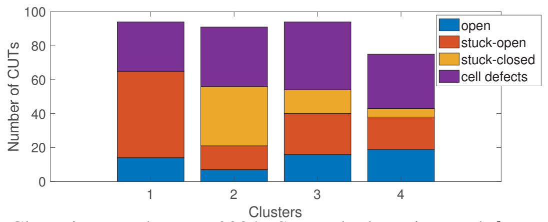

Fig. 6. Clustering results on s9234. Spectral clustering and $k$means ++ algorithm are employed on extracted features from the failing chips.

Fig. 7 shows several densely connected components. Two of them are denoted by A and B in the plot. Adjacent nodes indicate the corresponding failing chips are more correlated than the non-adjacent ones. Such correlation is verified as a function of defect types via hypothesis testing using permutation test. When choosing the failing individuals for more expensive and time-consuming PFA (physical failure analysis), it is desirable that the PFAed ones are representative of the entire population. Using Fig. 7 as an example, it would be reasonable to choose one from *A* component, one from *B* component, and one from the isolated nodes, if only three PFA sources are available.

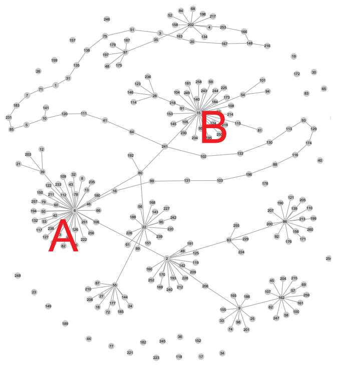

Fig. 7. Undirected graphical models from structural learning.

VI. Discussion and Conclusions

DIAGRAM differs from the early work in [3] in multiple aspects. First, the two works target entirely different problems. [3] aims to improve diagnosis results and identify the roots cause for the failing population. DIAGRAM uses the problem setting of [1] to evaluate its functionalities. Second, their model configurations (both observed and hidden node definition) are different. Finally, [3] produces no generative model, which means it can neither use missing data nor estimate model parameters.

DIAGRAM provides an opportunity of handling missing data by using generative models. Missing data exist for various reasons, in addition to the erroneous data logging. Some companies collect a finite amount of test data during ATE execution. The collection is terminated if the accumulated failing bits exceed the limit. As a result, the responses for the last failing pattern may be incomplete. On the other hand, information from physical layout and post-silicon circuit fabrics is usually very limited [24]. In any of the cases, working with partial or incomplete information is crucial for learning models in test and diagnosis applications.

Instead of parameter learning, DIAGRAM adopts structural learning to uncover the underlying correlation structure among the failing chips. The densely connected nodes are more likely to fail from similar causes. Since the primary objective of root cause analysis is to identify and understand the cause of most failures, we found DIAGRAM to be both promising and beneficial towards post-diagnosis failure analysis.

References

[1] H. Wang et al., "Test-data volume optimization for diagnosis," in *DAC*, 2012, pp. 567–572.
[2] Q. Huang, C. Fang, S. Mittal, and R. D. S. Blanton, "Improving diagnosis efficiency via machine learning," in *ITC*, 2018.
[3] B. Benware, C. Schuermyer, M. Sharma, and T. Hermann, "Determining a failure root cause distribution from a population of layout-aware scan diagnosis results," *IEEE Design & Test of Computers*, vol. 29, no. 1, pp. 8–18, 2012.
[4] L.-C. Wang and M. S. Abadir, "Data mining in EDA - basic principles, promises, and constraints," in *DAC (invited paper)*, 2014.
[5] J. Tikkanen, S. Siatkowski, N. Sumikawa, L.-C. Wang, and M. S. Abadir, "Yield optimization using advanced statistical correlation methods," in *ITC*, 2014.
[6] C.-K. Hsu et al., "Variation and failure characterization through pattern classification of test data from multiple test stages," in *ITC*, 2016.
[7] Y. Xue, X. Li, and R. D. Blanton, "Improving diagnostic resolution of failing ICs through learning," *TCAD*, vol. 37, no. 6, pp. 1288–1297, June 2018.
[8] H. G. Stratigopoulos, "Machine learning applications in IC testing," in *ETS*, 2018, pp. 1–10.
[9] C. Shan, P. Babighian, Y. Pan, J. Carulli, and L.-C. Wang, "Systematic defect detection methodology for volume diagnosis: A data mining perspective," in *ITC*, 2017, pp. 1–10.
[10] N. Sumikawa, L.-C. Wang, and M. S. Abadir, "A pattern mining framework for inter-wafer abnormality analysis," in *ITC*, 2013.
[11] C.-K. Hsu, P. Sarson, G. Schatzberger, F. Leisenberger, J. Carulli, S. Siddhartha, and K.-T. Cheng, "Variation and failure characterization through pattern classification of test data from multiple test stages," in *ITC*, 2016.
[12] A. Ahmadi, C. Xanthopoulos, A. Nahar, B. Orr, M. Pas, and Y. Makris, "Harnessing process variations for optimizing wafer-level probe-test flow," in *ITC*, 2016.
[13] Z. Poulos and A. Veneris, "Clustering-based failure triage for RTL regression debugging," in *ITC*, 2014.
[14] N. Sumikawa, M. Nero, and L.-C. Wang, "Kernel based clustering for quality improvement and excursion detection," in *ITC*, 2017.
[15] H. Wang and K. He, "Improving test and diagnosis efficiency through ensemble reduction and learning," *ACM TODAES*, vol. 24, no. 5, pp. 49:1– 49:26, 2019.
[16] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.
[17] K. P. Murphy, *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012.
[18] S. L. Lauritzen, *Graphical Models*. Clarendon Press, 1996.
[19] D. Koller and N. Friedman, *Probabilistic Graphical Models: Principles and Techniques*. The MIT Press, 2009.
[20] X. Ren, V. G. Tavares, and R. D. Blanton, "Detection of illegitimate access to JTAG via statistical learning in chip," in *DATE*, 2015, pp. 109–114.
[21] S. Biswas, H. Wang, and R. D. Blanton, "Reducing test cost of integrated, heterogeneous systems using pass-fail test data analysis," *ACM TODAES*, vol. 19, no. 2, pp. 20:1– 20:23, 2014.
[22] H.-G. D. Stratigopoulos, P. Drineas, M. Slamani, and Y. Makris, "Non-RF to RF test correlation using learning machines: a case study," in *VTS*, 2007, pp. 9–14.
[23] S. Biswas, P. Li, R. D. Blanton, and L. T. Pileggi, "Specification test compaction for analog circuits and mems," in *DATE*, 2005, pp. 164–169.
[24] Y. Xue, X. Li, R. D. Blanton, C. Lim, and M. E. Amyeen, "Diagnostic resolution improvement through learning-guided physical failure analysis," in *ITC*, 2016, pp. 1–10.
[25] H. H. Chen, R. Hsu, P. Y. Yang, and J. J. Shyr, "Predicting system-level test and in-field customer failures using data mining," in *ITC*, 2013.
[26] C. J. Shan, P. Babighian, Y. Pan, J. Carulli, and L.-C. Wang, "Systematic defect detection methodology for volume diagnosis: A data mining perspective," in *ITC*, 2017.
[27] S. Venkataraman and S. B. Drummonds, "POIROT: a logic fault diagnosis tool and its applications," in *ITC*, 2000, pp. 253–262.
[28] W. Zhang, X. Li, F. Liu, E. Acar, R. A. Rutenbar, and R. D. Blanton, "Improving diagnostic resolution of failing ICs through learning," *TCAD*, vol. 30, no. 12, pp. 1814–1827, December 2011.
[29] C. K. Chow and C. N. Liu, "Approximating discrete probability distributions with dependence trees," *IEEE Transactions on Information Theory*, vol. 14, no. 3, pp. 462–467, 1968.
[30] W. C. Tam and R. D. Blanton, "SLIDER: Simulation of layout-injected defects for electrical responses," *TCAD*, vol. 31, no. 6, pp. 918–929, 2012.
[31] S. Hojsgaard, "CRAN Task View: gRaphical Models in R," 2019. [Online]. Available: https://CRAN.R-project.org/view=gR
[32] WinBUGS, 2019. [Online]. Available: http://www.mrc-bsu.cam.ac.uk/software/bugs/the-bugs-project-winbugs/
[33] OpenBUGS, 2015. [Online]. Available: http://www.openbugs.net
[34] D. Lunn, C. Jackson, N. Best, A. Thomas, and D. Spiegelhalter, *The BUGS Book: A Practical Introduction to Bayesian Analysis*. Boca Raton, FL: Chapman and Hall/CRC, 2012.