Bayesian Concepts in Software Testing: An Initial Review

Daniel Rodriguez
Dept. of Comp. Science
University of Alcalá
Alcalá de Henares, 28871,
Madrid, Spain
daniel.rodriguezg@uah.es

Javier Dolado
Univ. Basque Country
UPV/EHU
Donostia-San Sebastián,
20080, Spain
javier.dolado@ehu.eus

Javier Tuya
Dept. of Comp. Science
University of Oviedo
Campus of Gijón, 33204,
Gijón, Spain
tuya@uniovi.es

ABSTRACT

This work summarizes the main topics that have been researched in the area of software testing under the umbrella of "Bayesian approaches" since 2010. There is a growing trend on the use of the so-called Bayesian statistics and Bayesian concepts in general and software testing in particular. Following a Systematic Literature Review protocol using the main digital libraries and repositories, we selected around 40 references applying Bayesian approaches in the field of software testing since 2010. Those references summarise the current state of the art and foster better focused research.

So far, the main observed use of the Bayesian concepts in the software testing field is through the application of Bayesian networks for software reliability and defect prediction (the latter is mainly based on static software metrics and Bayesian classifiers). Other areas of application are software estimation and test data generation. There are areas not fully explored beyond the basic Bayesian approaches, such as influence diagrams and dynamic networks.

Categories and Subject Descriptors

A.1 [**Introductory and Survey**]; D.2.5 [**Software Engineering**]: Testing and Debugging

General Terms

Theory

Keywords

Bayesian statistics, probabilistic graphical models, Bayesian networks, software testing

1. BAYESIAN CONCEPTS FOR SOFTWARE TESTING: AN INITIAL REVIEW

There have been several articles in the past advocating the use of Bayesian methods in software testing. The position paper by Namin and Sridharan [27] stated that Bayesian reasoning methods have the capability of improving the field of software testing by providing solutions based on probabilistic methods. However, Namin and Sridharan discuss three obstacles faced when applying Bayesian networks: (i) the generalization of the conclusions, (ii) the sensitivity to the prior probabilities and (iii) the difficulties for software engineers to grasp the statistical concepts underlying the Bayesian approach. The first two obstacles will be further discussed in Section 4 (Discussion) after reviewing the literature. With respect to the last obstacle, the hurdles of understanding the Bayesian concepts, the Bayesian approach takes a different viewpoint in the concept of probability from the frequentist approach [37], which could make it hard to understand:

- *Frequentists*: the definition of probability is related to the frequency of an event. The parameters of interest are fixed but the data are a repeatable random sample, hence there is a frequency. No prior information is used. In a strict frequentist view, it does not make sense to talk about the true value of the parameter θ under study. The true value of θ is fixed, by definition.
- *Bayesians*: the definition of probability is related to the level of knowledge about an event. The value of knowledge about an event is based on prior information and the available data. The parameters of interest are unknown and the data are fixed. From a Bayesian viewpoint we can talk about the probability that the true value of the parameter θ lies in an interval.

The fact that Bayesians use prior information about θ makes the statistical reasoning different. Given a set of data observations represented by D, we can compute $P(D|\theta)$ (θ is fixed) in the frequentist approach, but we can compute $P(\theta|D)$ in the Bayesian approach. $P(\theta|D)$ is computed using "Bayes' theorem":

$$\Pr(\theta|D) = \frac{\Pr(D|\theta)\Pr(\theta)}{\Pr(D)}. \quad (1)$$

Equation (1) contains the elements of the Bayesian inference process: $Pr(\theta|D)$ is the posterior probability, $\Pr(D|\theta)$ is the likelihood, $Pr(\theta)$ is the prior probability and $\Pr(D) = \Pr(D|\theta)\Pr(\theta)+\Pr(D|\neg\theta)\Pr(\neg\theta)$ is a normalizer factor. Thus, equation 1 allows us in this case to compute the probability of θ given D.

Bayes' theorem enables the computation of the posterior probabilities for a variable. A Bayesian Network (BN) is

A-TEST'15, August 30-31, 2015, Bergamo, Italy
© 2015 ACM. 978-1-4503-3813-4/15/08...$15.00
http://dx.doi.org/10.1145/2804322.2804329

a probabilistic graphical model with variables linked by directed arcs. A BN provides a way of modelling a domain problem using graph and probability theory, where the network representation of the problem can be used to generate information about some variable, provided that the information of its parents is available. The joint probability distribution for variables $X_1, X_2, \ldots, X_n$ can be calculated as follows:

$$P(X_1, X_2, \ldots, x_n) = \prod_{i=1..n} P\left(X_i | Parents(X_i)\right) \quad (2)$$

where $Parents(X_i)$ denote the specific values of the nodes in the graph that are linked towards the node X_i in the BN. BNs are the main use of the Bayes approach. BNs are also known as Bayesian Belief Networks or Belief Networks, which are Probabilistic Graphical Models. There are other formalisms related to the basic Bayesian approach such as Dynamic Belief Networks, Influence Diagrams and Hidden Markov Models.

2. SYSTEMATIC LITERATURE REVIEW OF BAYESIAN NETWORKS IN SOFTWARE TESTING SINCE 2010

Next, we review relevant literature related to software testing (including quality) and Bayesian concepts according to the Systematic Literature Review (SLR) protocol and guidelines suggested by Kitchenham et al. [19] and the EBSE website[1] [12].

2.1 Background and Research Aim

Recent works reviewing Bayesian networks (BN) include a position paper by Namin [27] and a review by Misirli and Basar [26] which covers several issues of decision making including defect prediction. Misirli and Basar list seven articles in "software testing" using the Bayesian approach and further 54 works under the topic of "software quality". The authors included under "software quality" the concepts of fault, failures, defect prediction and software reliability. However, there is an increasing number of papers applying Bayesian concepts in general and in testing in particular. Therefore, in this initial review our research aim is to *identify, classify and analyse the available literature since 2010 related to different aspects of software testing and quality that apply Bayesian concepts.*

2.2 Data Sources

We have used the following digital libraries and repositories as data sources:

1. ISI Web of Science (`http://apps.webofknowledge.com/`)
2. Scopus (`http://www.scopus.com/`)
3. Elsevier Science Direct (`http://www.sciencedirect.com/`)
4. IEEE Xplore (`http://ieeexplore.ieee.org/`)
5. SpringerLink (`http://www.springerlink.com/`)
6. ACM Digital Library (`http://www.acm.org/`)
7. Wiley Interscience (`http://onlinelibrary.wiley.com/`)
8. Google Scholar (`http://scholar.google.com/`)
9. The Collection of Computer Science Bibliographies (`http://liinwww.ira.uka.de/bibliography/`) Later discarted due to a large and irrelevant number of papers returned.

These generic digital libraries and repositories, together, reference all relevant publications in the software engineering field. Some of them are digital libraries, but some others are *meta-repositories*, including most relevant journals, conference and workshop proceedings.

2.3 Search Strategy

The search strategy was conducted with the following keywords in the query: "Bayesian" & "networks" & "software testing". We considered that those keywords were enough to cover all articles of interest. We tested other combinations of potential keywords but the results did not conform to the research aim. For example, we did not find relevant literature using the terms *probabilistic graphical models* but not containing *Bayes* or *networks* at the same time. Similarly, all papers about testing are almost certain to contain the substring "software testing". Finally, we followed some references from selected papers confirming that all relevant literature was found.

Table 1 shows the digital libraries used, the search domain within the repository (when possible) and the number of papers found. We merged all the references found and decide whether the paper was eligible for this survey mainly based on the title and abstract with the exception of Google scholar due to the large number of references found. Google Scholar was mainly used for checking that no important and relevant paper was missing. If a paper had an apparently relevant title or abstract, its full content was also checked to decided whether the study was to be included in the final selection.

2.4 Selection Criteria

The selection criteria consists of inclusion and exclusion criteria for the papers found. We include papers published since 2010, related to software testing, written in English and accessible on the Web. We decided to cover in this review the literature after the position paper of Namin and Sridharan [27]. There is a growing number of articles including the topics of software testing and Bayesian methods. Also, this criterion limits the number of papers to the most recent research and limits the length of this conference paper. A more comprehensive SLR is part of our current work.

The exclusion criteria include papers published before 2010. We also exclude papers related to the application of Bayesian concepts as part of some kind of optimisation in relation to other methods (e.g. forming part of neural networks). Also, some documents and articles that we have excluded, mentioned BNs without dealing with the technique. There were some articles that developed different tests based on BNs, but not strictly related to software testing, e.g., they developed "software safety tests". We leave out of review the use of neural networks that use some kind of Bayesian optimisation and other articles that are not fully focused on Bayesian

[1]Evidence-Based Software Engineering `http://www.dur.ac.uk/ebse/`

Table 1: Repositories and papers found and selected

Digital Library	*Domain*	*Returned*	*Relevant*
ISI Web of Science	Computer Science, Engineering	10	6
Scopus	Computer Science	45	16
Elsevier ScienceDirect	Computer Science	40	6
SpringerLink	-	133	10
IEEExplore	-	24	3
ACM DL	-	12	-
Wiley Interscience	-	62	-
Google Scholar	-	2,390	-
Total			41

procedures. For example, and interesting work by Wiper et al. [40] use Bayesian concepts for providing the prior distribution probabilities to an artificial neural network. Strictly speaking, this work uses Bayesian concepts but does not use Bayesian networks.

In the next Section we analyse the selected literature grouping the different subareas and discussing possible research paths.

3. RESEARCH TOPICS ADDRESSED

We have organised the research topics into categories classified by common subdisciplines in software testing: effort testing prediction, software reliability and fault prediction, quality models, test data generation, GUI testing and a less known but interesting subarea named philosophy of technology.

3.1 Software Testing Effort Prediction and Productivity Estimates

This topic is concerned with the estimation of the test costs in terms of person-hours or person-days. Few works have recently applied Bayesian models for testing effort estimation, exceptions include the works by Torkar et al. [36], Schulz et al. [33] and Dalmazo et al. [8] which describe a defect correction effort model. A generic survey about BNs in effort estimation, including the test phase, was carried out by Radlinski [30].

3.2 Fault and Defect Prediction. Software Reliability

The topic of reliability is another area where Bayesian approaches have been explored by multiple researchers, specially for real-time systems. "Software Reliability" is the probability that software will work without failing in a specified environment for a given amount of time. Software reliability testing tries to discover as many defects as possible as early as possible.

Defect prediction from static measures from private or open repositories such as Tera-Promise[2] (formerly known as the Promise repository) have been reported on multiple studies. Bayesian classifiers have been widely used. Recent examples include the work by Dejaeger et al. [9] who compared 15 BN classifiers for the task of identifying software faulty modules. Weyuker et al. [39] also performed a comparison of tools for fault prediction that included Bayesian additive regression trees. Another comparison of classifiers including BNs and Naïve Bayes is described in Dhankhar et al. [10]. The Naïve Bayes classifier, the simplest Bayesian approach, is extensively used before and after 2010. For example, we can cite the papers by Catal et al. [5], Ma et al. [25] and by Hewett [15] for comparing the approaches to software defect prediction.

Okutan and Yildiz [28] used Bayesian networks to explore the relationship between sets of metrics and defect proneness using datasets from the Promise repository. Other works that build Bayesian networks with predictive reliability are those of Cheng-Gang et al. [6], Kumar and Yadab [20], Abreu et al. [1], Jongsawat and Premchaiswadi [16], Li and Wang [22], Rekab et al. [31], Lv et al. [24], Blackburn and Huddell [4], Qiuying et al. [29], Jun-min et al. [17], Khan et al. [18], Li and Leung [21], Cotroneo et al. [7], Ba and Wu [3] and Zheng et al. [43]. An application of software reliability with BNs in the domain of fire control radar can be found in the work by Li et al. [23].

3.3 Quality Models

A quality model describes in a structured way the concept of quality in a software system. In this category, we found the work by Wagner [38] who considers software quality based on constructing a BN from an activity-based quality model. Schumann et al. [34] also describe a Bayesian Software Health Management system in which the reliability of a system, including software and hardware, is monitored with BNs.

3.4 Test Data Generation, Test Case Selection and Test Plan Generation

Test data generation and test case priorization are important areas within software testing. A recent work by Sagarna et al. [32] explore this path as part of search based software test data generation. The improvement of random testing has been tackled by Zhou et al. [44, 45]. Sridharan and Namin reported [35] on the priorization of mutation operators. Several experiments were carried out by Do et al. [11] concerning the priorization of test cases. The authors used BNs as one of the methods for ordering the test suites. Fang and Sun [13] proposed a strategy to optimize the re-execution of test cases (regression testing) based on BNs. Finally, Han [14] built a BN by converting a Fault Tree structure of events in order to perform forward and backward reasoning.

3.5 Graphical User Interface (GUI) Testing

Another area of recent application of BNs is GUI testing. Yang et al. [41, 42] built a BN that uses the prior knowledge of testers and the BN updates the values depending on the

[2]`http://openscience.us/repo/`

results of the test cases.

3.6 Philosophy of Technology

As an *outlier* paper, we were positively surprised by the recent work by Angius [2] where the Bayes concepts and the software testing field have been used as the substrate for defining the software engineering area as a "scientifically attested technology". This paves the way for more studies relating the disciplines of software testing and the philosophy of technologies.

4. DISCUSSION

The main use of the Bayesian concepts in software testing lie on the "software reliability" area, with 60% of the publications falling in this category. Other topics of applications are "test data generation" and "test effort estimation", with 11% and 10% of the references, respectively. Topics where BNs are not so extensively used were "quality models", "GUI testing" and "philosophy of technology".

Although there is an increasing number of works applying BN approaches, there are issues that hinder their application as previously mentioned such as their steep learning curve and problems related to the statistical analyses. With respect to these two problems (also previously mentioned in the introduction) and discussed by Namin and Sridharan [27], we may highlight the following issues, after reviewing the literature:

- *Generalization of the conclusions*: every work builds its BN starting from scratch and the BN is adapted to its specific problem. A "meta study" or meta-analysis of the results obtained by different researchers would uncover potential similarities in the results and in the graphical structure of the BN.
- *Sensitivity to priors*: an essential characteristic of BNs is the need to provide prior probabilities to variables. One way to avoid discrepancies is to set standard priors in the field, which could be agreed upon in case of parameters such as productivity, etc. However, it is not always possible to agree on priors nor the non-informative priors are adequate to the BN model. Other alternatives could include the use of hyperprior distributions. The fact that BNs allow us to update the variable probabilities can moderate the results obtained with different priors, provided a robust BN.

Probabilistic graphical models can help in testing activities (and decision making in general) as supervised (prediction) and unsupervised (clustering) techniques from the data mining point of view as well as optimisation approaches. In prediction, we can consider classifiers such as Naïve Bayes and more complex structures such as TAN (Tree Augmented Naïve Bayes) to generic networks such as Bayesian Networks or Markov Models and their extensions (e.g. Dynamic BNs, Influence Diagrams). These latter Bayesian approches have not yet been fully exploited (in comparison with the former simpler Bayesian classifiers).

In the case of graphical models for optimisation, Evolutionary Distribution Algorithms have been applied successfully in software testing (although mainly prior to 2010). These approaches have also been applied to data generation which is considered to be a preprocessing step in data mining and, in our opinion, they can be further explored.

5. CONCLUSIONS

In this work, we reviewed the recent literature on probabilistic graphical models in software testing. We found around 40 references dealing with the topics of interest since 2010. The spread of topics found within the software testing area applying BNs is fairly limited. We classified the references into six categories. The fact that the main category is related to "software reliability" may distort the potential applications of BNs to other areas in software testing. Interestingly, there was a reference that positioned the concept of "software testing" in the center of study of software engineering as a science.

As our current work, we are extending this systematic literature survey.

6. ACKNOWLEDGMENTS

The authors are grateful to the reviewers and chairs for their helpful comments. Partial support has been received by Project Iceberg FP7-People-2012-IAPP-324356 (D. Rodriguez) and Project TIN2013-46928-C3.

7. REFERENCES

[1] R. Abreu, A. Gonzalez-Sanchez, and A. J. Van Gemund. A diagnostic reasoning approach to defect prediction. In *Modern Approaches in Applied Intelligence*, pages 416–425. Springer, 2011.

[2] N. Angius. The problem of justification of empirical hypotheses in software testing. *Philosophy & Technology*, 27(3):423–439, 2014.

[3] J. Ba and S. Wu. Propred: A probabilistic model for the prediction of residual defects. In *Mechatronics and Embedded Systems and Applications (MESA), 2012 IEEE/ASME International Conference on*, pages 247–251. IEEE, 2012.

[4] M. Blackburn and B. Huddell. Hybrid bayesian network models for predicting software reliability. In *2012 IEEE Sixth International Conference on Software Security and Reliability Companion*, 2012.

[5] C. Catal, U. Sevim, and B. Diri. Practical development of an eclipse-based software fault prediction tool using naive bayes algorithm. *Expert Systems with Applications*, 38(3):2347 – 2353, 2011.

[6] B. Cheng-Gang, J. Chang-Hai, and C. Kai-Yuan. A reliability improvement predictive approach to software testing with bayesian method. In *Control Conference (CCC), 2010 29th Chinese*, pages 6031–6036, July 2010.

[7] D. Cotroneo, R. Natella, and R. Pietrantuono. Predicting aging-related bugs using software complexity metrics. *Performance Evaluation*, 70(3):163 – 178, 2013. Special Issue on Software Aging and Rejuvenation.

[8] B. L. Dalmazo, A. L. R. de Sousa, W. L. Cordeiro, J. Wickboldt, R. C. Lunardi, R. L. dos Santos, L. P. Gaspary, L. Z. Granville, C. Bartolini, M. Hickey, et al. It project variables in the balance: a bayesian approach to prediction of support costs. In *Software Engineering (SBES), 2011 25th Brazilian Symposium on*, pages 224–232. IEEE, 2011.

[9] K. Dejaeger, T. Verbraken, and B. b. Baesens. Toward comprehensible software fault prediction models using

bayesian network classifiers. *IEEE Transactions on Software Engineering*, 39(2):237–257, 2013.

[10] S. Dhankhar, H. Rastogi, and M. Kakkar. Software fault prediction performance in software engineering. In *Computing for Sustainable Global Development (INDIACom), 2015 2nd International Conference on*, pages 228–232, March 2015.

[11] H. Do, S. Mirarab, L. Tahvildari, and G. Rothermel. The effects of time constraints on test case prioritization: A series of controlled experiments. *IEEE Transactions on Software Engineering*, 36(5):593–617, 2010.

[12] EBSE. Template for a systematic literature review protocol. `http://www.dur.ac.uk/ebse/resources/templates/SLRTemplate.pdf`, 2010.

[13] Z. Fang and H. Sun. A software regression testing strategy based on bayesian network. In *Computational Intelligence and Software Engineering (CiSE), 2010 International Conference on*, pages 1–4. IEEE, 2010.

[14] L. Han. Evaluation of software testing process based on bayesian networks. In *Computer Engineering and Technology (ICCET), 2010 2nd International Conference on*, volume 7, pages V7–361. IEEE, 2010.

[15] R. Hewett. Mining software defect data to support software testing management. *Applied Intelligence*, 34(2):245–257, 2011.

[16] N. Jongsawat and W. Premchaiswadi. Developing a bayesian network model based on a state and transition model for software defect detection. pages 295–300, 2012.

[17] Y. Jun-min, C. Ying-xiang, C. Jing-ru, and F. Jia-jie. Optimization model of software fault detection. In *Proceedings of the 2012 International Conference on Information Technology and Software Engineering*, pages 129–136. Springer, 2013.

[18] G. Khan, S. Sengupta, and K. Das. A probabilistic model for analysis and fault detection in the software system: An empirical approach. *Lecture Notes in Electrical Engineering*, 298 LNEE:253–265, 2014.

[19] B. A. Kitchenham, T. Dyba, and M. Jørgensen. Evidence-based software engineering. In *Proceedings of the 26th International Conference on Software Engineering (ICSE'04)*, pages 273–281, Washington, DC, USA, 2004. IEEE Computer Society.

[20] C. Kumar and D. Yadav. Software defects estimation using metrics of early phases of software development life cycle. *International Journal of System Assurance Engineering and Management*, pages 1–9, 2014.

[21] L. b. Li and H. Leung. Bayesian prediction of fault-proneness of agile-developed object-oriented system. *Lecture Notes in Business Information Processing*, 190:209–225, 2014.

[22] Q. Li and J. Wang. Determination of software reliability demonstration testing effort based on importance sampling and prior information. *Advances in Intelligent and Soft Computing*, 126 AISC:247–255, 2012.

[23] Z. Li, Q. Zhao, C. Li, and X. Yang. Design and reliability evaluation of simulation system for fire control radar network. In *International Conference on Quality, Reliability, Risk, Maintenance, and Safety Engineering (ICQR2MSE)*, pages 1314–1317, June 2012.

[24] J. Lv, B.-B. Yin, and K.-Y. Cai. Estimating confidence interval of software reliability with adaptive testing strategy. *Journal of Systems and Software*, 97(0):192 – 206, 2014.

[25] Y. Ma, G. Luo, X. Zeng, and A. Chen. Transfer learning for cross-company software defect prediction. *Information and Software Technology*, 54(3):248 – 256, 2012.

[26] A. T. Misirli and A. B. Bener. Bayesian networks for evidence-based decision-making in software engineering. *IEEE Transactions on Software Engineering*, 40(6):1–1, 2014.

[27] A. S. Namin and M. Sridharan. Position paper: Bayesian reasoning for software testing. *Proc. of the FSE/SDP workshop on Future of software engineering research (FoSER '10)*, 2010.

[28] A. Okutan and O. T. Yıldız. Software defect prediction using bayesian networks. *Empirical Software Engineering*, 19(1):154–181, 2014.

[29] L. Qiuying, L. Haifeng, and W. Guodong. Sensitivity analysis on the influence factors of software reliability based on diagnosis reasoning. In *Intelligence Computation and Evolutionary Computation*, pages 557–566. Springer, 2013.

[30] L. Radlinski. A survey of bayesian net models for software development effort prediction. *International Journal of Software Engineering and Computing*, 2(2):95–109, 2010.

[31] K. Rekab, H. Thompson, and W. Wu. A multistage sequential test allocation for software reliability estimation. *IEEE Transactions on Reliability*, 62(2):424–433, 2013.

[32] R. Sagarna, A. Mendiburu, I. Inza, and J. A. Lozano. Assisting in search heuristics selection through multidimensional supervised classification: A case study on software testing. *Information Sciences*, 258(0):122 – 139, 2014.

[33] T. Schulz, L. Radlinski, T. Gorges, and W. Rosenstiel. Defect cost flow model. In *Proceedings of the 6th International Conference on Predictive Models in Software Engineering - PROMISE '10*, page 1, New York, New York, USA, Sept. 2010. ACM Press.

[34] J. Schumann, T. Mbaya, O. Mengshoel, K. Pipatsrisawat, A. Srivastava, A. Choi, and A. Darwiche. Software health management with Bayesian networks. *Innovations in Systems and Software Engineering*, 9(4):271–292, June 2013.

[35] M. Sridharan and A. S. Namin. Prioritizing mutation operators based on importance sampling. In *Software Reliability Engineering (ISSRE), 2010 IEEE 21st International Symposium on*, pages 378–387. IEEE, 2010.

[36] R. Torkar, N. M. Awan, A. K. Alvi, and W. Afzal. Predicting software test effort in iterative development using a dynamic bayesian network. In *21st IEEE International Symposium on Software Reliability Engineering*. IEEE, 2010.

[37] J. VanderPlas. Frequentism and bayesianism: A python-driven primer. *arXiv preprint arXiv:1411.5018*, 2014.

[38] S. Wagner. A Bayesian network approach to assess

and predict software quality using activity-based quality models. *Information and Software Technology*, 52(11):1230–1241, Nov. 2010.

[39] E. J. Weyuker, T. J. Ostrand, and R. M. Bell. Comparing the effectiveness of several modeling methods for fault prediction. *Empirical Software Engineering*, 15(3):277–295, 2010.

[40] M. Wiper, A. Palacios, and J. Marın. Bayesian software reliability prediction using software metrics information. *Quality Technology & Quantitative Management*, 9(1):35–44, 2012.

[41] Z. Yang, Z. Yu, and C. Bai. The approach of graphical user interface testing guided by bayesian model. *Lecture Notes in Electrical Engineering*, 277 LNEE:385–393, 2014.

[42] Z.-F. Yang, Z.-X. Yu, B.-B. Yin, and C.-G. Bai. Gui reliability assessment based on bayesian network and structural profile. *International Journal of Signal Processing, Image Processing and Pattern Recognition*, 8(1):225–240, 2015.

[43] C. Zheng, F. Peng, J. Wu, and Z. Wu. Software life cycle-based defects prediction and diagnosis technique research. In *Computer Application and System Modeling (ICCASM), 2010 International Conference on*, volume 8, pages V8–192. IEEE, 2010.

[44] B. Zhou, H. Okamura, and T. Dohi. Markov chain monte carlo random testing. In *Advances in Computer Science and Information Technology*, pages 447–456. Springer, 2010.

[45] B. Zhou, H. Okamura, and T. Dohi. Enhancing performance of random testing through markov chain monte carlo methods. *IEEE Transactions on Computers*, 62(1):186–192, 2013.