Structural Learning Techniques for Bayesian Attack Graphs in Cyber Physical Power Systems

Abhijeet Sahu
Texas A&M University, College Station
abhijeet_ntpc@tamu.edu

Katherine Davis
Texas A&M University, College Station
katedavis@tamu.edu

***Abstract*—Updating the structure of attack graph templates based on real-time alerts from Intrusion Detection Systems (IDS), in an Industrial Control System (ICS) network, is currently done manually by security experts. But, a highly-connected smart power systems, that can inadvertently expose numerous vulnerabilities to intruders for targeting grid resilience, needs automatic fast updates on learning attack graph structures, instead of manual intervention, to enable fast isolation of compromised network to secure the grid. Hence, in this work, we develop a technique to first construct a prior Bayesian Attack Graph (BAG) based on a predefined threat model and a synthetic communication network for a cyber-physical power system. Further, we evaluate a few score-based and constraint-based structural learning algorithms to update the BAG structure based on real-time alerts, based on scalability, data dependency, time complexity and accuracy criteria.**

***Index Terms*—Bayesian Network, Bayesian Attack Graph, Structural Learning**

I. Introduction

Since power systems are critical infrastructure, grid resilience to cyber threats is a priority for national security. Ongoing and historical events including the Ukranian attack in 2015, the Stuxnet attack in 2008, and even the latest Mumbai power outage attack in October 2020 [1] motivate the criticality of grid cybersecurity. Cyber threats and propagation of their impacts can be analyzed by performing cyber-security risk assessment [2], where such assessments benefit from the use of attack graphs. Attack graphs are used to analyze network and host vulnerabilities as well as potential access paths adopted by adversaries to exploit these vulnerabilities to compromise their target [3].

Graphical models are used in the cyber-physical systems (CPS) community to allow both visual interpretations and algorithmic techniques to analyze and infer vulnerabilities. Graphical formalisms such as Bayesian networks (BNs) are versatile due to their ease of construction which can include information from cybersecurity domain experts, threat models, and even from raw data learning. Because an adversary's behavior is uncertain, BNs as Probabilistic Graphical Models (PGM) are beneficial for modeling attack graphs. In this work, we hypothesize that BNs can be useful in the cyber-physical power system setting to perform causal reasoning between each step in the access paths of the adversary's trajectory toward the compromise of its final target. A challenging task is to determine the structure of these BNs from raw data, e.g., from network logs or Intrusion Detection Systems (IDS). Hence, this paper addresses this challenge, where our contributions are as follows: **a.** *We create synthetic BAGs from the threat models on a synthetic electric power and communication network based on the BN's parameters, including the conditional probability densities from the Common Vulnerability Scoring System (CVSS) scores from National Vulnerability Database (NVD).* **b.** *We implement and compare structure learning algorithms for the synthetic BAGs.* **c.** *From analysis of the structure learning techniques applied to BAG, we infer the most suitable techniques.*

The paper proceeds as follows. Section II provides a review of the usage of PGM in CPS. Section III introduces the notion of attack graphs and the creation of BAGs. In Section IV, we present the types of structure learning techniques considered in this work. Algorithm the construct prior BAG and to learn structure is introduced in Section V. Finally, in Section VI, we evaluate the performance of the learning methods for synthetics BAGs based on size, computation time, data dependency, and accuracy. Section VII concludes the paper.

II. Background

BNs can be used for modeling critical infrastructures and their interdependencies. For example, a coupled intrastructure is modeled to study the impact of cascading effects between a power grid and its communication systems using BNs [4]. Similarly, a causal polytree based anomaly reasoning engine is proposed in [5] that utilizes Bayesian inference to produce a high-level view of the security state of a SCADA network. However, these works consider BNs for inferring posterior probabilities of intrusions, not for the problem of learning the BN structures that is tackled here.

BNs have been predominantely used in the areas of network security through construction of attack graphs. In [6], BNs are used to build an automatic and adaptive IDS for attack signature recognition. In [2], a dynamic BN for risk assessment is proposed, since the network state changes in real-time. BNs also have applications for model-based power system diagnosis [7] and decision support [8]. Contrary to the above works, our work on learning BNs is not restricted to pure network security or physical systems, but expanded to interdependent networks.

Numerous works examine automatic attack graph generation. Researchers in [9] propose two novel approaches for generating attack privilege fields as prerequisites and post-conditions from the NVD. Authors in [10] propose a practical approach to construct knowledge graphs from structured data, while [11] proposes an automatic generation algorithm for penetration graphs that optimizes the network topology before generating the penetration graph. Instead of using NVD scores to approximate attack characteristics, our work focuses on constructing a Bayesian variant of attack graphs for cyber-physical power systems.

Attack graphs are commonly constructed by security experts, but such templates depend on the expert's assumptions and cannot capture zero-day exploits. To solve this problem, alerts generated from IDS can act as a data source for the structure learning problem to learn the structure of an attack graph based on the prior structure provided by experts. Learning techniques can be classified into two types: score-based and constraint-based. Constraint-based methods use statistical evaluation to learn conditional independence from the data and prune the graph-searching using the obtained constraints,

2021 IEEE Texas Power and Energy Conference (TPEC) | 978-1-7281-8612-2/21/$31.00 ©2021 IEEE | DOI: 10.1109/TPEC51183.2021.9384933

while a score-based algorithm assigns each candidate Directed Acyclic Graph (DAG) a score reflecting its quality of fit, which is further optimized. A BN comprises a DAG, where nodes are the random variables, and conditional probability distributions (CPD) are defined for variables given their parent nodes. Learning the graph structure of these networks from data is NP-hard [12]. Approximate procedures can handle larger networks but usually get stuck in local maxima. The authors in [12] use score-based learning to learn BN structures from data based on score functions such as the Bayesian Information Criterion. A constraint-based learning example based on d-separation, is given in [13]. The above procedures were developed as a general mathematical tool for BN structure learning, but not evaluated for AGs or BAGs in particular.

The authors in [14] provide a review of various dynamic structure learning techniques, where the structure is updated based on new data received. Additional review papers, such as [15], [16], and [17] comprehensively review different state-of-the-art structure learning methods by considering discrete and continuous random variables in the BN. However, these works were not evaluated for BAGs for CPS on the basis of accuracy, computation time, data dependency, and scalability. Measuring *computation time* assists in faster incident response, *accuracy* improves corrective actions, and *data dependency* helps in selecting a learning technique based on availability of trustworthy alerts. Hence, this work evaluates the performance of different score-based and constraint-based structure learning algorithms, which will assist a security analyst to adaptively respond as the system evolves dynamically.

III. Bayesian Attack Graph (BAG)

A. Attack Graph (AG)

Attack graphs represent how the vulnerabilities in a network can be sequentially or parallely exploited, showing diverse paths an adversary can take to get to its targeted victim. AGs can be state-based or logical [18]. In state-based AGs, each node represents a unique security state within the network, where a combination of hosts is compromised. Scalability issues exist here, particularly when there is increased network density due to higher connectivity. Logical AGs represent dependencies, e.g., AND and OR, between exploits and security conditions.

ICS attack graph: This scenario is based on a cyber topology that mimics the control center of the Iranian Nuclear plant compromised by the Stuxnet worm (Fig. 1). The Enterprise Control Network has the $WinCC$, $Historian$, and $PCS7$ web clients. $WinCC$ is a SCADA-HMI system, and $PCS7$ is a Distributed Control System by Siemens. These clients access the web server through the $Web\ Navigator$, $PCS7\ Web\ Server$, and $CAS\ Server$ in the Demilitiarized Zone (DMZ). These web servers interact with process control network components such as $Engineering\ Workstation$, $Simatic\ Process\ Historian$, $PCS7\ OS$ which may control PLCs, relays, or controllers for circuit breakers, isolators, etc.

The configurations of Access Control Lists (ACLs) in the two firewalls, the vulnerabilities in the web servers in the DMZ, and the probable attack target, e.g., getting root access to $WinCC$ or $Eng.\ Workstation$ are used to create the logical AG (Fig. 1). The security conditions such as F : $Win\ CC\ Server$ represented as the circle nodes, meaning the intruder from $Web\ Nav\ Server$ get the privilege now over the host $Win\ CC\ Server$ by exploiting the vulnerability $Network\ Share(D, F)$ represented as rectangle nodes. The probability of the exploitation of the vulnerability to reach a security condition is assumed to be based on the CVSS scores associated with the vulnerability.

B. Bayesian Attack Graph (BAG)

Making use of a Bayesian Network (BN) on the AG makes it a BAG. A BN is a directed graphical model, with the nodes represented through random variables and the inter-dependencies between them is captured through the directed edges, forming a DAG [18]. Assuming the monotonicity principle of an AG, where once an attacker escalate a privilege never relinquishes it, one can remove duplicated paths to construct a DAG, hence it is suitable to model AGs as BAGs. The BAG in the rightmost figure of Fig. 1, have a joint probability distribution as:

$$\begin{aligned} p(A,B,C,D,E,F,G,H) = p(A)p(B)p(C \mid A) \\ p(E \mid C)p(D \mid B)p(F \mid D,B)p(G \mid D)p(H \mid E,F,G) \end{aligned} \tag{1}$$

The BN is first introduced for the dynamic analysis of attack graphs in [19], where the attacker's probability to reach a security state is computed, given the prior probability. A node in the BAGs is represented by a security condition, e.g. $Privilege(A)$, shown in the rightmost side in Fig. 1. The Conditional Probability Table (CPT) of the BAG is computed as the combined effect of vulnerability in a network, as used in [20]. The local CPT of the nodes with logical AND and OR conditions in the BAG are computed as Eqn. 2 [18]. A logical AND signifies all the permissives suffice to compromise the node X_i. A logical OR signifies any one permissive is sufficient to compromise a node. The immediate parents of the node X_i is $\mathbf{pa_i}$ and the probability that the vulnerability v_j is exploited is p_{v_j}.

$$\begin{aligned} p\left(X_i|\mathbf{pa}_i\right) &= \begin{cases} 0, & \exists X_j \in \mathbf{pa}_i | X_j = 0 \\ \prod_{j:X_j} p_{v_j}, & \text{otherwise} \end{cases} \\ p\left(X_i|\mathbf{pa}_i\right) &= \begin{cases} 0, & \forall X_j \in \mathbf{pa}_i | X_j = 0 \\ 1 - \prod_{j:X_j} \left(1 - p_{v_j}\right), & \text{otherwise} \end{cases} \end{aligned} \tag{2}$$

C. Application in Cyber Physical Testbed

A large scale synthetic power system [21], its communication network, and its associated threat models are utilized to synthesize AGs. The AGs generated in our previous works [22], [23] were non-Bayesian in nature. This work generates BAGs from the cyber-physical models validated in RESLab testbed [24]. Further, this work seeks to provide a verdict on preferring a specific algorithm over another, given the data in the form of alerts.

We generate synthetic clustered BAGs, with each cluster associated with one substation. The size of each cluster N depends on the number of vulnerabilities and the number of intruder entry points in each substation. For example, a power network with three substations having two vulnerabilities and one entry points each, will have a cluster size of at least 3 nodes, with a total of four clusters (three substations plus the main control center), resulting in a total of 12 nodes in the complete synthetic BAG. Fig. 2 shows a synthetic BAG of a substation and its control center with a cluster size of 10 each, resulting in a 20-node BAG. This work analyzes the impact of the graph size and density on computation time and accuracy in learning structures in an ICS network. The dynamic update of the structure based on alerts will help update access paths to critical assets such as PLCs, relay controllers, etc. These updates on access paths will enable dynamic situational awareness of the CPS.

IV. Structural Learning

Now that we have introduced the basics of attack graphs, BAG and its application in a CPS testbed, we briefly present

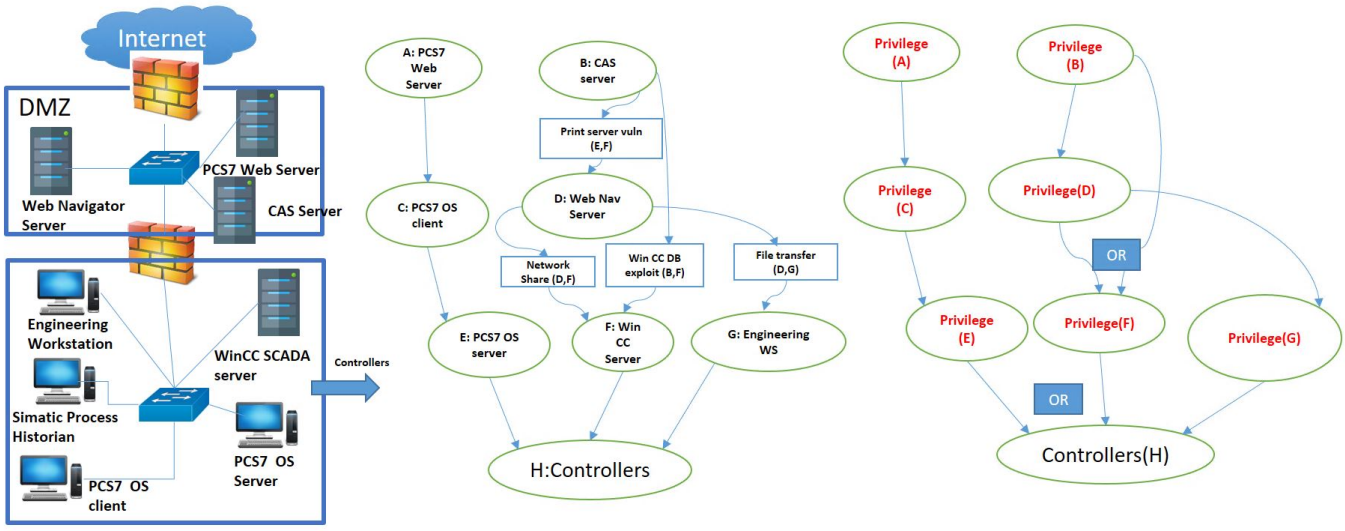


Fig. 1. SCADA network to Attack Graph (AG) to Bayesian Attack Graph (BAG) modeling

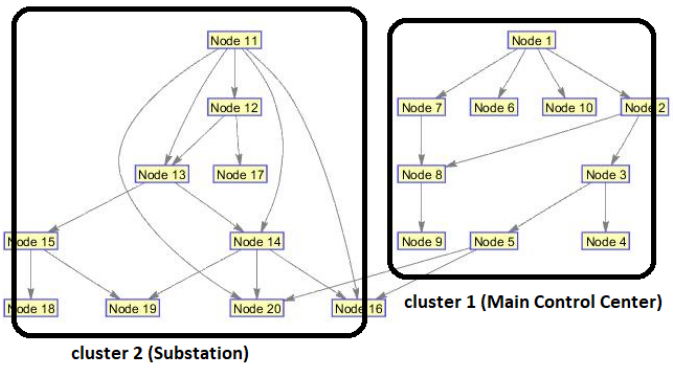


Fig. 2. Synthetic BAG generated for a substation and control center with 20 nodes with cluster size (N = 10).

the learning algorithm considered for the evaluation on the synthetic BAGs. We evaluate three score-based algorithm: Cooper and Herskovits K2 [13], Monte Carlo Markov Chain (MCMC) [25], Chow Liu [26] and a constraint-based algorithm: Partially Directed Acyclic Graph (PDAG) PC [27].

A. Cooper and Herskovits K2 Algorithm

A Bayesian Network is defined through its structure, B_s and parameterized by B_p. The K2 algorithm compares the joint distribution of $P(B_s, D)$, for all possible B_s and B_p combination. D here refers to the distribution of the data generated. The objective of the K2 algorithm is to maximize the joint distribution for all possible B_s as:

$$\max_{B_s}\left[P\left(B_s, D\right)\right] = \prod_{i=1}^{n}\max_{\pi_i}\left[P\left(\pi_i \rightarrow x_i\right)\prod_{j=1}^{q_i}\frac{(r_i-1)!}{(N_{ij}+r_i-1)!}\prod_{k=1}^{r_i}N_{ijk}!\right] \quad (3)$$

where, π_i is the parents node of x_i, q_i is the unique instantiation of π_i, r_i is the possible values for random variable x_i, N_{ijk} is the number of data points in D in which a variable x_i equals v_ik and π_i instantiated as w_{ij}, and $N_{ij} = \sum_{k=1}^{n_i} N_{ijk}$. The above maximisation operation (Eq. 3) is performed using a heuristic search method. The algorithm initiates with a parentless node and then incrementally add parents, whose addition improves the probability of the resulting structure the most and stops when cannot further increase the probability. But finding the most likely belief network structure is computationally feasible under the assumptions of node ordering and existence of tight limits on the number of possible parents. In the synthetic BAG, the topology ordering of the nodes can be ensured, but the existence of tight limits on the number of parents cannot be ensured as for a larger network there can have many vantage points for an intruder to target preventing the parent size for any node.

The time complexity of this method is $O(mn^2u^2r)$, where u is the maximum number of allowable parents of a variable. In the current work we will study how the u impacts the computation time. Higher the number of parents, it implies the attacker has more paths to compromise a node. So we want to study how a more vulnerable network increases the computation time of determining the structure of the BAGs. We also study the impact of the number of datapoints m considered for learning the structure.

B. Monte Carlo Markov Chain (MCMC) Algorithm

The Bayesnet library makes use of the paper [27], for performing MCMC learning. It uses the Metropolis-Hastings method, which constructs a Markov chain with the objective of maximizing the posterior distribution $p(g|x)$, where x is given data. The method comprises of evaluating at each step in the Markov chain, whether a candidate model g' replace the current model g with an acceptance probability.

This method first makes all the digraphs that differ from the initial G_o by a single edge deletion, addition or reversal operation, with acyclicity constraint [27]. Further with every

new sample in the markov chain, it updates the digraph from g to g' with the probability of $min(1, R_a)$, where R_a is

$$R_a = \frac{\#(nbd(g))p\,(g'|x)}{\#\,(nbd\,(g'))\,p(g|x)} \tag{4}$$

where, $\#(nbd(g))$ refers to the cardinality of collection of graph that can be reached from g in one step and the ratio of $\frac{p(g'|x)}{p(g|x)}$ is the Bayes Factor computed using the likelihood ratios, which varies for addition, removal and reversal operations [27]. The steps involved consists of *proposing* a g' and *moving* to g' from g processes. The computation time for *proposing* for the addition operation is $O(1)$ and $O(n)$ for the reversal operation, while the computation time for *moving* for the addition operation is $O(n)$ and atmost $O(n^2)$ for removal [27]. Unlike the K2 algorithm the time complexity is independent of the maximum number of allowable parents hence its performance will not vary with the complexity of the network. Hence, learning for the denser 8 substation model [23] would not be quite high in comparison to the sparse IEEE 24 bus cyber model [23]. Sample size increases computation time, hence more trustworthy samples is essential.

C. Chow Liu Algorithm

The Chow Liu algorithm is based on computing the mutual information (H) between the pair-wise features in the data-set.

$$H(U, V) = \sum_{u,v} \hat{p}(u, v) \log \left[\frac{\hat{p}(u, v)}{\hat{p}(u)\hat{p}(v)} \right] \tag{5}$$

Then utilizing this H as the weight, it constructs a Maximum Weight Spanning Tree (MWST) using Kruskal's algorithm. This tree is the Chow Liu Tree. There have been numerous variations of the algorithm since 1968 implementation, but they are not considered in current implementation. The Chow-Liu Algorithm has a time complexity of $O(n^2)$, as it takes $O(n^2)$ to compute H for all pairs, and $O(n^2)$ to compute the MWST. The Bayesnet library of MATLAB didnt include the Chow Liu algorithm for structural learning. Hence, it is incorporated in MATLAB using the algorithm in [26].

D. Partial DAG PC Algorithm

In some scenarios, it may not be possible to learn a DAG, instead we learn a partial DAG or PDAG. A PDAG is an acyclic graph containing both directed and undirected edges. The PC (Peter and Clark) algorithm is based on the d-separation test of conditional independence [13]. This algorithm basically consist of two steps: it first identifies the skeleton, then learns the *complete PDAG*. Learning using this algorithm assumes all variables to be observed. The output P is an adjacency matrix, in which $P(i, j) = -1$, if there is an $i \rightarrow j$ edge. $P(i, j) = P(j, i) = 1$ if there is an undirected edge $i \leftrightarrow j$ [28]. This algorithm may take $O(n^u)$ time if there are n variables and u is the maximum fan-in or allowable parents. One major disadvantage of this method is, even for the sparse networks, the algorithm gets infeasible with increasing number of nodes.

V. Pseudo-code of the Implementation

Alg. 1 provides the pseudo-code to construct the DAG, the function $ConstructDAG()$ and convert it to a BAG by building the CPT for each node $GetCPD()$, then finally learn the structure using different learning methods. The rightmost figure in Fig. 1 can be divided into two clusters, with each cluster having 3 and 4 nodes each. Hence, we generate the synthetic DAG based on the number of clusters C and nodes per cluster as N. Sim_Count is the number of simulation for a unique configuration, dag is the adjancency matrix representing the graph, etc. Alg. 1 shows the learning using $learn\_struct\_K2$. Similarly, other learning algorithms are tested by calling a different function.

The $ConstructDAG()$ in Alg. 2 creates the clustered DAG by randomly picking edges, keeping the maximum parent nodes to be u. The updated dag_mod is used to update the dag structure for each cluster and finally dag is used to create links between clusters. The $GetCPD()$ constructs the conditional probability distribution of each node based on the CVSS scores and Eqns. 2 utilized in lines 3 and 4 of Alg. 3. The source codes for the BAG generation and structure learning are available at [29].

Algorithm 1 BAG Structure Learning Implementation

1: Define C,N,Sim_Count,dag,p_{OR}, $samples$.
2: **for** u = 2 to 5 **do**
3: **for** sim = 1 to 20 **do**
4: $dag = ConstructDAG(u, C, N, dag)$ ▷ Create DAG using Alg. 2 passing attributes u, C, N, dag
5: $bnet = mk\_net(dag, C * N, node\_type)$ ▷ Create a BN $bnet$ from DAG using the mk_net of Bayesnet
6: **for** i = 1 to $C * N$ **do** ▷ for each node in the BN
7: $bnet.CPD(i) = CreateCPD(p_{OR}, dag, i, bnet)$ ▷ Obtain the CPD table for each node using Alg. 3
8: **end for**
9: $data = zeros(C * N, samples)$
10: **for** m = 1 to $samples$ **do**
11: $data(m) = sample\_bnet(bnet)$ ▷ Create data samples for each node in the BN
12: **end for**
13: $infer\_dag = learn\_struct\_K2(data)$ ▷ Learn the structure. For example, this is learning using K2 algorithm
14: $accuracy = compare(dag, infer\_dag)$ ▷ compare the edges between actual dag with $infer\_dag$
15: **end for**
16: Compute $avg\_accuracy$ and avg_comp_time
17: **end for**

Algorithm 2 $ConstructDAG(u, C, N, dag)$

1: **for** j = 1 to C **do**
2: Define dag_mod for cluster j
3: **for** i = 2 to N **do**
4: Create edges randomly for each node i in cluster j having maximum number of parents u
5: Update dag_mod
6: **end for**
7: Update dag for cluster j from dag_mod
8: Create random edges between any node in the cluster j and the rest of the clusters.
9: Update dag with new edges.
10: **end for**
11: **return** dag

Algorithm 3 $GetCPD(p_{OR}, dag, bnet)$

1: $npa = sum(dag)$
2: $prob = getCVSS(npa)$ ▷ Extract probability based on CVSS
3: **if** $rand(1) \leq p_{OR}$ **then** $cpt = createORTable(prob)$
4: **else** $cpt = createANDTable(prob)$
5: **end if**
6: **return** $tabular\_CPD(bnet, i, cpt)$ ▷ Construct CPD table

VI. Results and Analysis

In this section, we present our experimental results comparing the performance and accuracy of the four learning techniques introduced, based on the network size, data dependency, computation time and accuracy.

A. Experimental setup

Experiments are carried out using synthetic CVSS scores and BAGs generated using the methods discussed in Section V. The number of nodes in attack graphs as well as the density is varied across different experiments to study the impact of inference algorithms. In every experiment, 20 simulations are performed for calculating the average computation time for each scenario. The maximum in-degree (input edges) of the node, u , is varied from 2 to 5, as in most cases, the causes of an attack have an in-degree in that range. For complex attacks, the in-degree may be high but usually an attacker is resource-limited. The experiments are performed using a PC with an i7 1.80 GHz processor and 16 GB RAM. In our experiments, the sampled data generated from the synthetic BAGs in line 11 of the Alg. 1 is based on discrete random variables that indicate if the privilege over a component is escalated or not. We can generate samples for water or power networks based on continuous random variables.

B. Analysis of Structural Learning

A prior BAG is constructed as illustrated in the previous section. Based on the prior BAG, a $sample\_bnet$ function in Bayesnet library is used to generate a random sample for each node. All the nodes in the BAG has a CPD defined based on the CVSS score. Hence, the $sample\_bnet()$ function generates samples based on the CPD, which are used for learning the structure. For the experiments, the number of data samples used is 1000 except for MCMC technique where the number od data samples is altered. To explore the scalability factor, the number of nodes in the BAGs are altered from N = 6 to 20 while evaluating the average computation time (c_t) and the accuracy (acc). The accuracy is computed by comparing the adjacency matrix of the original and the learned BAG.

1) Learning with K2 Algorithm: The time complexity using K2 algorithm is theoretically $O(mn^2u^2r)$, which can be validated from the Fig. 3(a), where the c_t increased from 0.05 sec to 0.6 sec for $n = 6$ to $n = 20$. When u is increased from $u = 2$ to $u = 5$, the acc reduces for all scales of network (Fig. 4(a)). Hence, we conclude that as the fan-in (or in-degree) increases, the learning accuracy decreases, but since the acc is above 80% hence, K2 is not bad for learning. But since, the method's time complexity is of the order of 2, for n, it takes a long time to learn a large graph. This technique can be utilized to infer the structure of attack graphs locally in a small network. Hence, K2 algorithm can be implemented for larger networks by segregating the network prior to learning.

2) Learning with MCMC Algorithm: The MCMC learning time complexity is dependent on the number of nodes as well as total and burn-in samples. We observed that with the increase in samples and graph density (represented through u), the c_t increased (Fig. 3(b)). Similar trends are observed for acc(Fig. 4(b)) but relatively low acc compared to K2 algorithm. Even the c_t is relatively higher in the case of MCMC i.e. 1000 samples it is around 2 sec (Fig. 3(b)), unlike K2 which is less than 0.1 sec (Fig. 3(a)). For case $u = 5$, where the acc improved when the number of samples increased from 1000 to 5000. Hence, MCMC cannot be preferred for techniques where there are not enough samples to estimate the structure. Therefore, it is suggested to deploy more number IDS with higher sensitivity (i.e. high probing rate) for learning using MCMC. In this method, there is a tradeoff between the c_t and acc. A higher acc for a dense network is ensured by taking more samples which results in higher c_t.

3) Learning with Chou Liu Algorithm: Similarly, the implemented Chow Liu algorithm is validated for 1000 samples with varying N and u. The u have less impact on the c_t of the algorithm, as it is an approximation algorithm which cannot capture the complete structure of the original distribution as it only consider second order interaction (i.e. mutual information between two features). The acc is affected negatively as the u increased (Fig. 4(c)). The time complexity seemed to follow the theoretical $O(n^2)$, but the value is too high in comparison to other methods (Fig. 3(c)). The last stage in the Chow Liu technique involves computation of the spanning tree. In our implementation we have used the Kruskal's algorithm using the graph toolbox of MATLAB, which may be one probable reason of high c_t.

4) Learning with PCAG PC Algorithm: Finally, the PDAG PC algorithm is tested for the synthetic BAG generated for different scenario. Similar trends are observed i.e. c_t increasing with n and u as observed from Fig. 5. It is difficult to compute the accuracy of this algorithm with the current Bayesnet implementation, which can be further performed in the future.

The comparative study based on the scale, data dependency, c_t and acc (ranked) is presented in Table I.

TABLE I
COMPARATIVE STUDY OF LEARNING TECHNIQUES BASED ON SCALE, DATA DEPENDENCY, AVG. COMPUTATION TIME, AND ACCURACY

Technique	Scale	Data Dep.	c_t	acc
Cooper and Herskovits (K2)	✓	✗	✓	✓(1)
MCMC	✗	✓	✗	✓(3)
Chou Liu	✗	✗	✗	✓(2)
PDAG PC	✗	✗	✓	N.A

VII. CONCLUSION

The major contributions of this paper are the generation of BAGs for CPS, integration of Chow Liu algorithm in Bayesnet alongwith the analysis and comparison of other structural learning algorithms based on scale, data dependency, time complexity, and accuracy for BAGs in the cyber-physical power systems. It is observed that among the score-based learning techniques, K2 algorithm outperforms the Chow Liu and MCMC techniques with respect to all factors. The Chow Liu computation can be improved by incorporating the latest extensions. The computation time of the constraint-based PDAG PC is comparable to the score-based techniques but found to be unscalable as well as difficult to evaluate the accuracy. Apart from these techniques, recently developed Graph Neural Networks for DAG structure learning [30] can be considered in the area of cyber security in power systems.

VIII. ACKNOWLEDGEMENTS

This research is supported by the US Department of Energy's (DoE) Cybersecurity for Energy Delivery Systems program under award DE-OE0000895.

REFERENCES

[1] Pierluigi Paganini. (2020, Nov) October Mumbai power outage may have been caused by a cyber attack. [Online]. Available: https://securityaffairs.co/wordpress/111209/hacking/mumbai-power-outage-cyber-attack.html

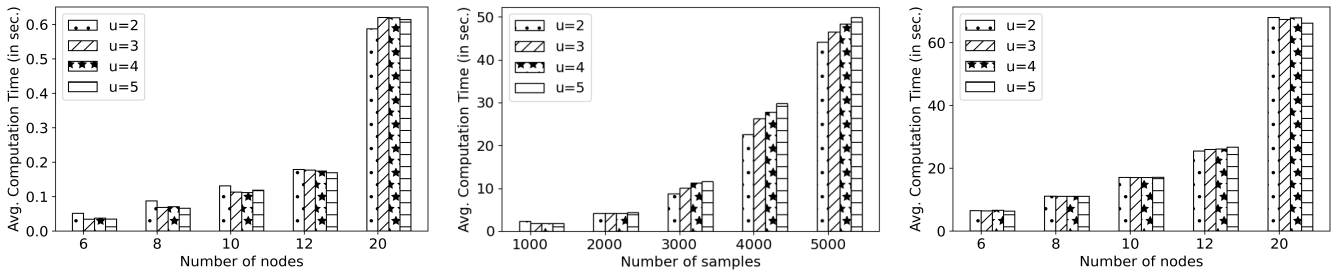
[2] J. Wang, K. Fan, W. Mo, and D. Xu, "A method for information security risk assessment based on the dynamic bayesian network," in *2016 International Conference on Networking and Network Applications*.

[3] S. Jha, O. Sheyner, and J. Wing, "Two formal analyses of attack graphs," in *Proceedings 15th IEEE Computer Security Foundations Workshop. CSFW-15*, 2002, pp. 49–63.

[4] N. HadjSaid, C. Tranchita, B. Rozel, M. Viziteu, and R. Caire, "Modeling cyber and physical interdependencies - application in ict and power grids," in *2009 IEEE/PES Power Systems Conference and Exposition*.

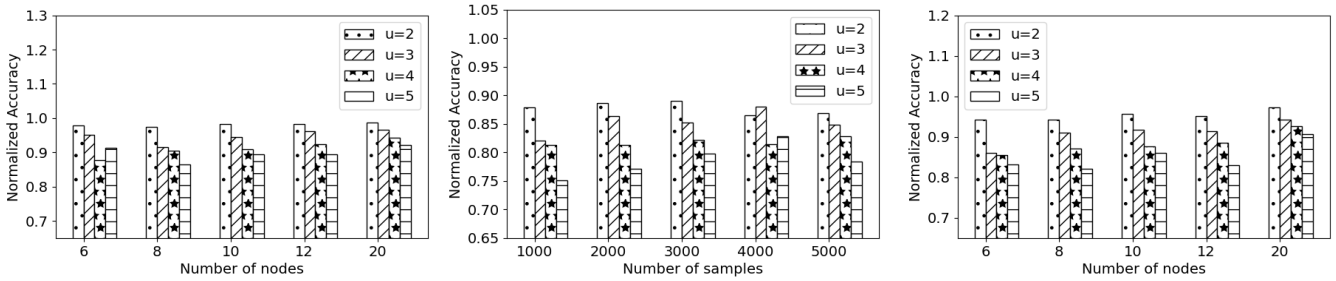
[5] W. Ren, T. Yu, T. Yardley, and K. Nahrstedt, "Captar: Causal-polytree-based anomaly reasoning for scada networks," in *2019 IEEE SmartGridComm*, 2019, pp. 1–7.

[6] F. Jemili, M. Zaghdoud, and M. B. Ahmed, "A framework for an adaptive intrusion detection system using bayesian network," in *2007 IEEE Intelligence and Security Informatics*, 2007, pp. 66–70.



(a) (b) (c)

Fig. 3. (a) Computation time for varying N (number of nodes) and u (number of maximum parent), using K2 algorithm; (b) Computation time for varying N (number of samples), with N=10, and u, using Improved MCMC algorithm; (c) Computation time for varying N and u, using Chow Liu Algorithm



(a) (b) (c)

Fig. 4. (a) Accuracy for varying N (number of nodes) and u (number of maximum parent), using K2 algorithm; (b) Accuracy for varying N (number of samples), with N=10, and u, using Improved MCMC algorithm; (c) Accuracy for varying N and u, using Chow Liu Algorithm

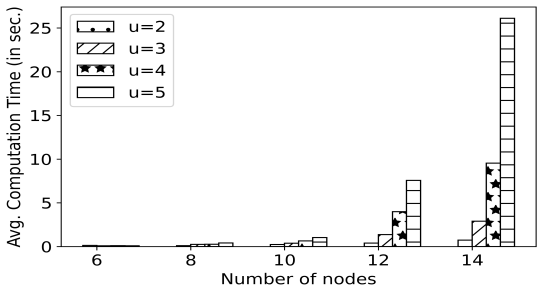


Fig. 5. Computation time with varying N and u using PDAG PC algorithm

[7] O. J. Mengshoel, M. Chavira, K. Cascio, S. Poll, A. Darwiche, and S. Uckun, "Probabilistic model-based diagnosis: An electrical power system case study," *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, vol. 40, no. 5, pp. 874–885, 2010.

[8] B. Delfino, G. B. Denegri, M. Invernizzi, A. Canonero, and P. Forzano, "An expert system for alleviating overloads in electric power systems: general concepts and applications," in *[1988] Proceedings. The Fourth Conference on Artificial Intelligence Applications*, 1988, pp. 299–304.

[9] M. Aksu, K. Bicakci, M. H. Dilek, M. Ozbayoglu, and E. Tatlı, "Automated generation of attack graphs using nvd," 2018.

[10] Y. Jia, Y. Qi, H. Shang, R. Jiang, and A. Li, "A practical approach to constructing a knowledge graph for cybersecurity," *Engineering*, vol. 4, no. 1, pp. 53 – 60, 2018, cybersecurity.

[11] Xueqiu, Q. Jia, S. Wang, C. Xia, and L. Lv, "Automatic generation algorithm of penetration graph in penetration testing," in *2014 Ninth International Conference on P2P, Parallel, Grid, Cloud and Internet Computing*, 2014, pp. 531–537.

[12] C. Campos and Q. Ji, "Efficient structure learning of bayesian networks using constraints," *Journal of Machine Learning Research*, 2011.

[13] P. Spirtes, C. Glymour, and R. Scheines, *Causation, Prediction, and Search*, 01 1993, vol. 81.

[14] L. Pereira, J. Nunes, M. Perkusich, K. Gorgonio, H. Almeida, and A. Perkusich, *Continuous Learning of the Structure of Bayesian Networks: A Mapping Study*, 11 2018.

[15] S. Beretta, M. Castelli, I. Gonçalves, R. Henriques, and D. Ramazzotti, "Learning the structure of bayesian networks: A quantitative assessment of the effect of different algorithmic schemes," *Complexity*, 04 2017.

[16] M. Scanagatta, A. Salmerón, and F. Stella, "A survey on bayesian network structure learning from data," *Progress in Artificial Intelligence*, pp. 1–15, 2019.

[17] B. Ellis and W. Wong, "Learning causal bayesian network structures from experimental data," *Journal of the American Statistical Association*, vol. 103, pp. 778–789, 02 2008.

[18] L. Muñoz-González, D. Sgandurra, M. Barrère, and E. C. Lupu, "Exact inference techniques for the analysis of bayesian attack graphs," *IEEE Transactions on Dependable and Secure Computing*, vol. 16, 2019.

[19] Y. Liu and H. Man, "Network vulnerability assessment using bayesian networks," *Proc SPIE*, 03 2005.

[20] M. Frigault, L. Wang, A. Singhal, and S. Jajodia, "Measuring network security using dynamic bayesian network," in *Proceedings of the 4th ACM Workshop on Quality of Protection*, ser. QoP '08. New York, NY, USA: Association for Computing Machinery, 2008.

[21] A. B. Birchfield, T. Xu, K. M. Gegner, K. S. Shetye, and T. J. Overbye, "Grid structural characteristics as validation criteria for synthetic networks," *IEEE Transactions on Power Systems*, 2017.

[22] K. R. Davis, R. Berthier, S. Zonouz, G. Weaver, R. B. Bobba, E. Rogers, and D. M. N. P. W. Sauer, "Cyber-physical security assessment for electric power systems," in *IEEE-HKN: The Bridge*, 2017.

[23] A. Sahu, H. Huang, K. Davis, and S. Zonouz, "A framework for cyber-physical model creation and evaluation," in *2019 20th International Conference on Intelligent System Application to Power Systems (ISAP)*.

[24] A. Sahu *et al.*, "Design and evaluation of a cyber-physical resilient power system testbed," 11 2020. [Online]. Available: http://arxiv.org/abs/2011.13552

[25] G. F. Cooper and E. Herskovits, "A bayesian method for the induction of probabilistic networks from data," *Machine Learning*, vol. 9, 1992.

[26] C. Chow and C. Liu, "Approximating discrete probability distributions with dependence trees," *IEEE Transactions on Information Theory*, vol. 14, no. 3, pp. 462–467, 1968.

[27] P. Giudici and R. Castelo, "Improving markov chain monte carlo model search for data mining," *Machine Learning*, pp. 127–158, 01 2003.

[28] "How to use the bayes net toolbox," Oct, 2007. [Online]. Available: http://bayesnet.github.io/bnt/docs/usage.html

[29] A. Sahu, "Cps bag codes for structure learning," https://github.com/Abhijeet1990/CPS_BAG.git, 2020.

[30] Y. Yu, J. Chen, T. Gao, and M. Yu, "Dag-gnn: Dag structure learning with graph neural networks," 2019.