Review of Inferencing and Structural Learning Techniques for Bayesian Attack Graphs

Abhijeet Sahu
Department of Electrical and Computer Engineering
Texas A&M University
College Station, TX 77801
abhijeet_ntpc@tamu.edu

Abstract

This is a review of exact and approximate inferencing of the posterior probabilities of the occurrence of a security escalation event in the Bayesian Attack Graphs. The impact of the structural learning problem based on accuracy and time complexity, based on the prior structure provided by the domain expert, and synthetic alerts generated from the Intrusion Detection System (IDS) is also reviewed. For the current work, we haven't considered actual alerts from the IDS, but used some synthetic alerts from a given probability distribution. The inferencing techniques explored in the current work are: Variable Elimination (12), Junction tree (11), Pearl's Belief Propagation Algorithm (13), Belief Propagation using factor graphs. (14). The structural learning techniques explored are Cooper and Herskovits K2 algorithm (8), MCMC (9), Chow Liu Algorithm (10), PDAG PC (7), etc. All the techniques were explored using the Bayes Net MATLAB Library by Kevin Murphy (1). The Chow Liu Algorithm for structural learning from data, is implemented in MATLAB.

1 Motivation

The cyber security risk assessment in any organisation is performed by the use of attack graphs. Attack graph analyses network and host vulnerabilities and the access path adopted by attackers to exploit these vulnerability to compromise their target (2). Bayesian networks are used to model these attack graphs due to two reasons. Firstly, since the attacker's behavior is uncertain so probabilistic graphical models are adopted. Secondly, Bayesian networks are used due to the existence of causal reasoning between each steps in the access paths of the attackers trajectory to compromise the target. Performing Bayesian Inferencing on these Bayesian Attack Graphs(BAG), can help has compute the likelihood of a host in the network is compromised or not. The overall goal of this project is to analyse the performance and accuracy of the Bayesian Network implementation for attack graphs that can be utilized for different scales of computer network to prevent cyber intrusions.

2 Background

Usage of Probabilistic graphical models for security analysis is a new area less explored. But there are couple of recent works who have made use of Bayesian Attack Graphs for causal reasoning among security incidents. The author's in (3), provide a framework for an adaptive intrusion detection system that uses Bayesian network. An exact inferencing algorithm using junction tree and belief propagation is proposed in (4) that uses synthetic BAG with different topologies to validate their algorithm on providing computational advantages over variable elimination technique used in inferencing. The authors in this work have adopted Shenoy-Shafer algorithm (5) for the junction tree computation followed by belief propagation. A causal polytree based anomaly reasoning engine is proposed in (6)

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

that utilizes Bayesian inference on the causal polytrees to produce a high-level view of the security state of the protected Supervisory Control and Data Acquisition (SCADA) network.

3 Attack Graphs (AG)

Attack Graphs are graphical models that represent the how vulnerabilities in a network can be sequentially or parallelly exploited, showing different paths an attacker can take to reach its target. In literature there are state-based and logical AGs (4). In state-based representation, each node represents

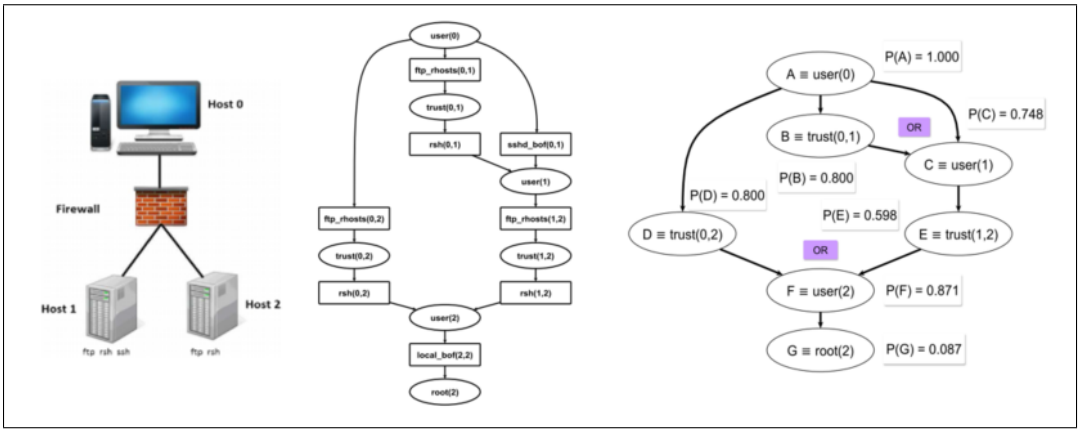


Figure 1: Computer network to attack graphs to Bayesian Attack Graph(BAG) modeling

collectively the security state of the network where a combination of a few host being compromised. There exist scalability issue with this approach, as the network becomes dense where each host has higher connectivity among each other. Logical based attack graphs represents dependencies between exploits and security conditions. In this work Logical attack graphs are studied.

3.1 Creation of Attack Graphs and Bayesian Attack Graphs

The scenario is adopted from the paper (4). The leftmost figure in Fig.1 represents a simple network where $Host1$ and $Host2$ running a few services such as File transfer Protocol (FTP), Secure Shell (SSH), Remote Shell (RSH) behind a firewall that allows traffic from external users lets say $Host0$. The Access Control Lists (ACLs) configuration in the firewall, the vulnerabilities in the servers, and the probable attack target, like getting root access to $Host2$ could be used to create the logical attack graph shown in the center of the figure. The security conditions such as $trust(0,1)$ are represented as the circle nodes (it means the $Host0$ has the potential to build $Host1$) and the vulnerabilities such as $rsh(1,2)$ (it means $Host1$ can exploit the remote shell vulnerability in $Host2$) are represented as rectangle nodes. The probability of the exploitation of the vulnerability to reach a security condition is based on the Base Score metric of the Common Vulnerability Scoring System (CVSS) score from National Vulnerbaility Database (NVD) (16). Assuming the monotonicity principle of the attack graph, where once an attacker escalate a privilege never relinquishes it, one can remove duplicated paths to construct a Directed Acyclic Graph. Since a DAG can be constructed, it is suitable to model AGs as Bayesian Attack Graphs. The Bayesian Network was first introduced for the dynamic analysis of attack graphs, where the probability that an attacker can reach a security state given prior probability of the state it has reached (4). The logical AND and OR conditions in the BAG are implemented using the Eqns. 1 and 2 as implemented in the paper (4).

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

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

4 Inferencing

Inferencing is done on the attack graphs to calculate the unconditional probability distributions to determine the probability that an attacker can reach a given security condition. So basically we will be performing marginalization. There are various methods for marginalization, out of which we will explore four techniques from Bayes Net toolbox.

4.1 Variable Elimination (12)

The variable elimination method as programmed in Bayes Net is basd on bucket elimination algorithm as per Fig.6 of the paper (12). The complexity of the algorithm is $O(nw*)$ where n is the number of nodes and $w*$ is the induced tree width. To address the exponential blow-up when computing marginal probabilities, it identifies factors in the joint distribution that depend on a small number of variables, computes them once and caches the results to avoid generating them exponentially many times '(4). The variables elimination order effects the size of the intermediate factors created impacting the computational complexity of the algorithm. Finding optimal elimination order is based on the concept of induced graph i.e. graph obtained when eliminating variable from the original one. The criterias usually followed for eliminating node to get induced graph are min-neighbours, min-fill, min-weight and weighted-min-fill (4). Using this method we need to compute the elimination order and the factors each time we make a query, whereas other algorithms like Belief Propagation or Junction Tree cache information (messages) that can be re-used to efficiently compute the marginal probabilities.

4.2 Junction Tree (11)

Junction tree makes use of the message passing algorithm as in belief propagation but is not only applicable for trees and polytrees. The Bayes Net library makes use of the the Shenoy-Shafer method which implements the same message passing scheme as BP. The method discussed in class is based on obtaining clique trees through making chordal graphs. The method employed in Bayes Net to obtain clique trees is based on variable elimination. The detailed example for the same can be found in (4). Considering discrete nodes, each having r values, Junction Tree scales in time and space as $O(|F|r^s)$ where $|F|$ is the number of factors and s is the size of the largest factor in the clique tree.

4.3 Pearl's Belief Propagation(BP) Algorithm and BP using Factor Graphs (13), (14)

If the BAG has no loops the Pearl's algorithm in the Bayes Net toolbox performs exact inferencing on the conditional probabilities, while it performs approximate inference on loopy graphs.

5 Learning

5.1 Structural Learning

Usually attack templates are constructed by security experts. Even though there exist such templates, there can be zero-day exploits which an expert cannot envision previously. Hence, alerts generated from the Intrusion Detection Systems (IDS) will act as the data source for the structure learning problem of learning the structure of attack graphs based on the prior structure provided by the experts with upgraded graphs based on the alerts (basically upgrading a few edges). In this regard we have explored three structure learning algorithms from the Bayes Net toolbox (8), (9), (7), etc as well as implemented Chow Liu algorithm (10).

5.1.1 Cooper and Herskovits K2 algorithm (8)

A Bayesian belief network is defined through its structure B_s and the parameter B_p. The K2 algorithm compares the joint distribution of $P(B_s, D)$, for all possible B_s possible for all possible B_p combination. The time complexity of computing $P(B_s, D)$ is $O(mn^2r + t_{Bs})$, where m is the number of data sets, r is the possible value each random variable can take, n is the number of random variable and t_{Bs} is the time required to compute the prior $P(B_s)$. The objective of the K2 algorithm

is to maximize the joint distribution for all possible B_s as given in Eq. 3.

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

where, π_i is the parents o x_i, q_i is the unique instantiation of π_i, r_i is the possible value random variable x_i can take, N_{ijk} is the number of data points in D in which a variable x_i takes the value $v_i k$ and π_i instantiated as w_{ij}, and $N_{ij} = \sum_{k=1}^{n_i} N_{ijk}$.

Finding most likely belief network structure is computationally feasible under the assumptions: a) there exist an ordering of the nodes, b) existence of tight limits on the number of possible parents of a node, c) $P\left(\pi_i \rightarrow x_i\right)$ and $P\left(\pi_j \rightarrow x_j\right)$ are marginally independent when i $\neq$ j.

The above maximisation operation is performed using a heurestic search method. The algorithm begins by making assumptions that a node has no parents and then incrementally adds parents whose addition most increases the probability of resulting structure. The loop stops when addition of no single parent can increase the probability. The time complexity of the K2 algorithm is found to be $O(mn^2u^2r)$ where u is the maximum number of allowable parent of a variable, with the worst case complexity being $O(mn^4r)$ where a node can have all the rest of the nodes as the parents. In the current work we will study how the u impacts the computation time of the algorithm. Higher the number of parents, it implies the attacker has more paths to compromise a node. So we want to study how more vulnerable network increases the computation time of determining the structure of the bayesian attack graphs. We would also study the impact of the number of datapoints m.

5.1.2 Monte Carlo Markov Chain (MCMC) (9)

The Bayes Net library makes use of the paper (9), for performing Monte Carlo Markov Chain based structural learning. This method is basically the Metropolis-Hastings method which builds a Markov chain that has the posterior distribution $p(g|x)$ as the objective distribution. 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.

The function first makes all the digraphs that differ from the initial G_o by a single edge deletion, addition or reversal operation, subject to the acyclicity constraint as described in the paper. Then in a for loop, for every sample in the markov chain, it makes use of the digraph neighbors to upgrade the current DAG g to new DAG g' and the ancestor matrix A, followed by computing the acceptance ratio R_a given by: $R_a = \frac{\#(nbd(g))p\left(g'|x\right)}{\#(nbd(g'))p(g|x)}$ in which the ratio of the posterior is equivalent to the likelihood ratio as per the Bayes rule, which is also termed as the Bayes Factor computed using $\frac{L\left(g',i\right)}{L(g,i)}$ for addition and removal operation, and $\frac{L\left(g',i\right)L\left(g',j\right)}{L(g,i)L(g,j)}$ for reversal operation, where the Likelihood $L(g,i)$ is given by Eq. 4.

$$L(g,i) = \prod_{j=1}^{q_i} \frac{\Gamma\left(N'_{ij}\right)}{\Gamma\left(N_{ij}+N'_{ij}\right)} \prod_{k=1}^{r_i} \frac{\Gamma\left(N_{ijk}+N'_{ijk}\right)}{\Gamma\left(N'_{ijk}\right)} \tag{4}$$

where, all the variables are same as defined for Eq. 3, except N'_{ijk} defines the prior hyperparameter. The steps involved in the algorithm consists of two processes, one is proposing a g' and another is moving to g' from g. The computation time for proposing for addition operation is $O(1)$ for addition and $O(n)$ for reversal, while the computation time for moving for addition operation is $O(n)$ and atmost $O(n^2)$ for removal.

5.1.3 Chow Liu Algorithm (10)

The Bayes Net library didnt include the Chow Liu algorithm for structural learning. Based on the method discussed in the class, I implemented the algorithm in MATLAB and compared its performance with the existing learning algorithm. The 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 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.

5.1.4 PDAG PC (7)

The PC algorithm is an extension to the Spirtes, Glymour, and Scheines (SGS) algorithm, which is based on the d-separation tests, can be found in page 82 of (7). It was found that even for sparse graphs the algorithm rapidly becomes infeasible as the number of vertices increases. Hence, they were looking for algorithm that has the same input/output relations as the SGS procedure for faithful distributions but which for sparse graphs does not require the testing of higher order independence relations in the discrete case(i.e. few d-separation tests).

The PC algorithm comprises of 3 steps: a) Forms the undirected graph C on the vertex list V. b) Run the loop until $Adj(C, X)\backslash Y$ defined as the set of vertices adjacent to A in the DAG C, is of the cardinality less than counter n in the loop. In the loop select the ordered pair (X, Y) such that $Adj(C, X)\backslash Y$ has cardinality greater than or equal to counter n and a subset S, where the pair X, Y are d-separated given S and record S in $Sepset(Y, X)$ and $Sepset(X, Y)$. c) Create the minimal pattern,i.e., the only directed edges with V structures. Convert the minimal pattern to a complete one,i.e., every directed edge in P is compelled and every undirected edge in P is reversible.

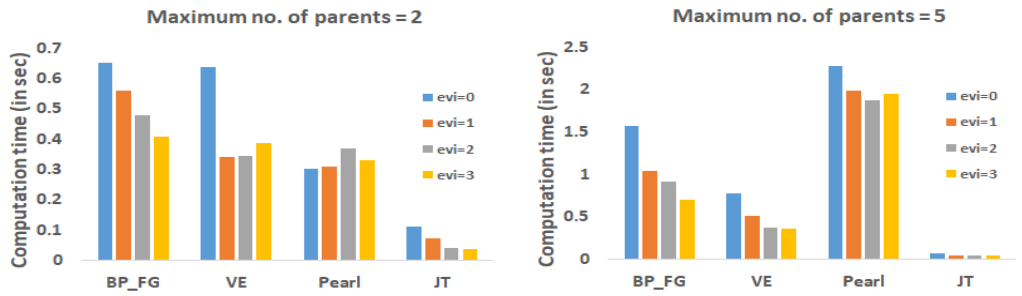
The PC algorithm does structure learning assuming all variables are observed. The output P is an adjacency matrix, in which $P(i, j) = -1$, if there is an $i- > j$ edge. $P(i, j) = P(j, i) = 1$ if there is an undirected edge $i < - > j$. This algorithm may take $O(n^k)$ time if there are n variables and k is the maximum fan-in.

6 Results and Analysis

In this section, first we present our experimental results comparing the performance (computation time) of different inference algorithm such as Belief Propagation using Factor Graphs, Variable Elimination, Junction Trees, Pearl's Belief Propagation, based on the impact of attack graph density and IDS detection capability. Further, we implement and compare the structural learning algorithm based on the computation and accuracy of the structure learned for MCMC, K2, PDAG and Chow Liu Algorithms. All the experiments were carried out using synthetic CVSS scores and bayesian attack graphs generated using the methods discussed in (4). The number of nodes in attack graphs as well as the density is varied across different experiments to study the efficiency of each algorithms. The accuracy in structural learning is evaluated based on the adjacency matrix (no. of matches in the edges in the prior and the learned structure). In every experiments number of simulations ran for calculating the average computation time and average accuracy is 20 for each scenario. The u in all the experiments is varied from 2 to 5 as most of the attacks causes have a fan-in in that range. For very very complex attacks the fan-in may be high but in a usual scenario an attacker may not have enough resources to be in that security state.

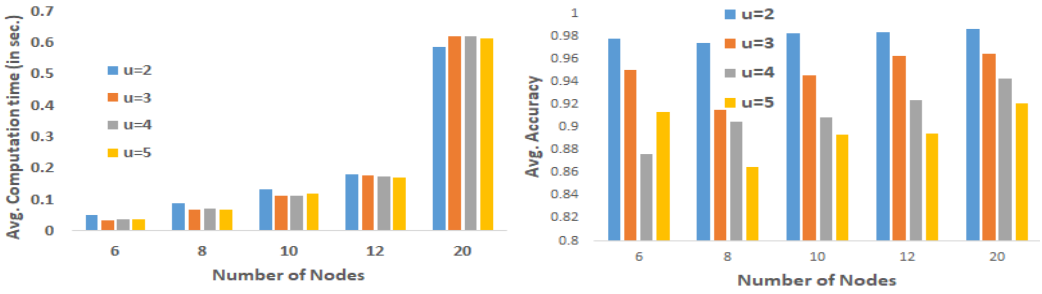
6.1 Analysis of Inferencing Algorithms on BAGs

For comparing the inference algorithm, we have generated synthetic graphs with a cluster structure (4), we have considered networks with clusters of the same size. For each cluster, we generate pseudo-random subgraphs with a maximum number of parents for each node to u. The Figs. 2(a) and 2(b) shows the average computation time using four inferencing algorithm with u=2 and u=5 respectively (3 cluster with cluster size of 10). We also altered the number of evidences from 0 to 3, which implies the number of observable nodes in the BAG (which implies number of nodes the IDS already sends an alert with 100 percent probability). It was observed that the Junction tree algorithm is faster in comparison to other methods. As the number of evidences increases the computation time for all the methods decreases. One important observation made was pearl's belief propagation time complexity increases as the network become dense. The Bayes Net toolbox implements both loopy and loop free implementation to do approximate and exact inferencing separately. Analysis on individual algorithm is performed, but due to space limitations the results are not presented in the report.



(a) (b)

Figure 2: (a) Comparison of inference techniques with maximum allowed parents (u = 2), with varying no. of observed nodes; (b) Comparison of inference techniques with maximum allowed parents (u = 5), with varying no. of observed nodes



(a) (b)

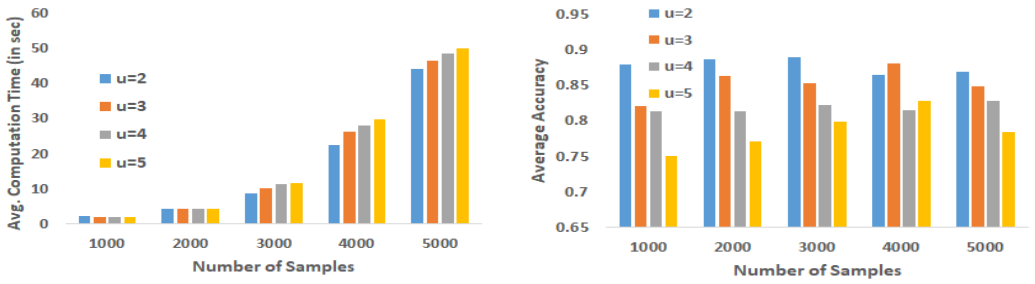
Figure 3: (a) Computation time for varying N (number of nodes) and u(number of maximum parent), using K2 algorithm; (b) Accuracy for varying N (number of nodes) and u(number of maximum parent), using K2 algorithm

6.2 Analysis of Structural Learning

For all the structural learning experiments number of data samples used is 1000 except for MCMC technique where analysis is performed on changing the number of data samples. The structural learning's time complexity using K2 algorithm was theoretical shown to increase to the power of 4 for the number of nodes and power of 1 for the number of data samples which can be practically validated from the Fig. 3(a). We found that as the u is increased, the accuracy of the method reduces for all different scales of network (n= 6 to 20) as observed in Fig. 3(b).

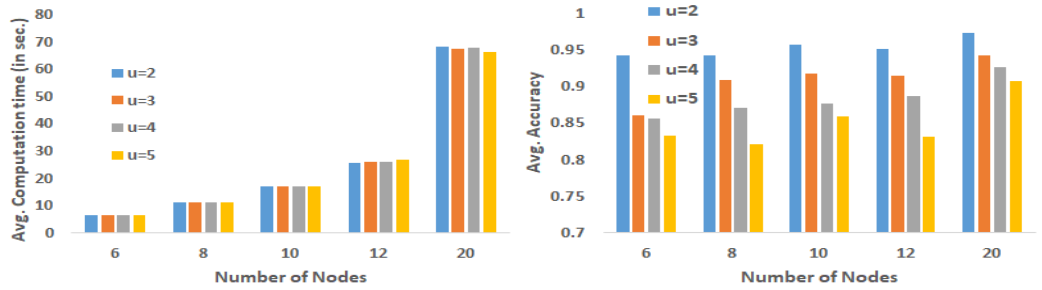
The structural learning's time complexity using MCMC algorithm is dependent on the number of nodes as well as the total and burn-in samples. We observed that with increase in samples and graph density (represented through u), the computation time of learning increased (Fig. 4(a)). Similar trends were observed for accuracy as can be seen from Fig. 4(b) but relatively low accuracy compared to K2 algorithm. Even the computation time is relatively higher in the case of MCMC algorithm (1000 samples it is around 2 sec(Fig. 4(a)), unlike K2 which is less than 0.1 sec(Fig. 3(a)))

Similarly the implemented Chow Liu algorithm was validated for 1000 data points with varying number of BAG nodes with varying u. The u did not have much impact on the time complexity 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. it only consider mutual information between two features). The accuracy were affected negatively as the u increased. The time complexity seemed to follow the theoretical $O(n^2)$ but the value was too high in comparison to other methods. The probable reason may be the time complexity of the default spanning tree algorithm (Kruskal's) used in Graph toolbox of MATLAB.



(a) (b)

Figure 4: (a) Computation time for varying N (number of samples), with N=10, and u(number of maximum parent), using Improved MCMC algorithm; (b) Accuracy for varying N (number of samples), with N=10, and u(number of maximum parent), using Improved MCMC algorithm



(a) (b)

Figure 5: (a) Computation time for varying N (number of nodes) and u(number of maximum parent), using Chow Liu Algorithm; (b) Accuracy for varying N (number of nodes) and u(number of maximum parent), using Chow Liu Algorithm

Finally, the learning algorithm for the PDAG PC algorithm is tested for the synthetic BAG generated for different scenario. Similar trends were observed i.e. computation time increasing with n and u as observed from Fig. 6.2.

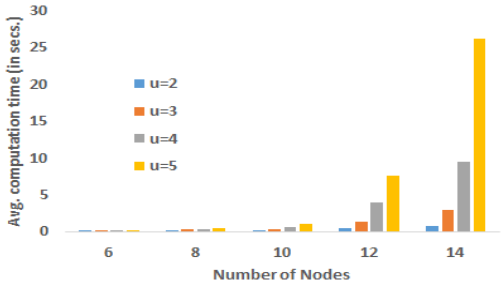


Figure 6: Computation time for varying N (number of nodes) and u(number of maximum parent), using PDAG PC algorithm

7 Conclusion and Scope of Future Work

The major contributions of this project are generation of BAGs, analysis and comparison of inference and structural learning algorithms based on time complexity and accuracy for different scales and densities of BAGs using Bayes Net Matlab toolbox as well as the implementation of Chow Liu structural learning algorithm in MATLAB.

In future, we will be implementing these probabilistic graphical models on attack graph templates from industries as well as intrusion detection alerts logs from a real testbed. Since, the Bayesnet library of the MATLAB is thoroughly studied for the methods described in the report, I expect to incorporate more recent algorithms in it and publish the work in some cyber-security conference.

References

[1] How to use the Bayes Net Toolbox, `http://bayesnet.github.io/bnt/docs/usage.html`

[2] S. Jha, O. Sheyner and J. Wing, "Two formal analyses of attack graphs," Proceedings 15th IEEE Computer Security Foundations Workshop. CSFW-15, Cape Breton, NS, Canada, 2002, pp. 49-63. doi: 10.1109/CSFW.2002.1021806

[3] F. Jemili, M. Zaghdoud and M. B. Ahmed, "A Framework for an Adaptive Intrusion Detection System using Bayesian Network," 2007 IEEE Intelligence and Security Informatics, New Brunswick, NJ, 2007, pp. 66-70. doi: 10.1109/ISI.2007.379535

[4] L. Muñoz-González, D. Sgandurra, M. Barrère and E. C. Lupu, "Exact Inference Techniques for the Analysis of Bayesian Attack Graphs," in IEEE Transactions on Dependable and Secure Computing, vol. 16, no. 2, pp. 231-244, 1 March-April 2019. doi: 10.1109/TDSC.2016.2627033

[5] Prakash P. SHENOY, Glenn SHAFER, Axioms for Probability and Belief-Function Propagation, Machine Intelligence and Pattern Recognition, North-Holland, Volume 9, 1990, Pages 169-198, ISSN 0923-0459, ISBN 9780444886507

[6] Wenyu Ren, Tuo Yu, Tim Yardley, Klara Nahrstedt, “CAPTAR: Causal-Polytree-based Anomaly Reasoning for SCADA Networks”, IEEE Smart Grid Communications (SmartGridComm), October 2019.

[7] Peter Spirtes Clark Glymour Richard Scheines, 2001. "Causation, Prediction, and Search, 2nd Edition," MIT Press Books, The MIT Press, edition 1, volume 1, number 0262194406

[8] Cooper and Herskovits, "A Bayesian method for the induction of probabilistic networks from data", Machine Learning Journal 9:308–347, 1992.

[9] P. Giudici and R. Castelo,"Improving MCMC model search for data mining", submitted to J. Machine Learning, 2001.

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

[11] "Probabilistic networks and expert systems", Cowell, Dawid, Lauritzen and Spiegelhalter, Springer, 1999

[12] R. Dechter, "Bucket Elimination: A Unifying Framework for Probabilistic Inference", UA1 96, pp. 211-219.

[13] Jonathan S. Yedidia, William T. Freeman, and Yair Weiss. 2003. Understanding belief propagation and its generalizations. In Exploring artificial intelligence in the new millennium, Gerhard Lakemeyer and Bernhard Nebel (Eds.). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA 239-269

[14] Kschischang, Frank R., Frey, Brendan J., Loeliger, Hans-Andrea (2001), "Factor Graphs and the Sum-Product Algorithm", IEEE Transactions on Information Theory, 47 (2): 498–519, doi:10.1109/18.910572, retrieved 2008-02-06.

[15] K. Murphy, Y. Weiss and M. Jordan, "Loopy belief propagation for approximate inference: an empirical study", UAI 99

[16] National Vulnerbality Database, `https://nvd.nist.gov/vuln-metrics/cvss`