Data Analytics for Fault Localization in Complex Networks

Maggie X. Cheng and Wei Biao Wu

***Abstract*—We consider the problem of identifying the source of failure in a network after receiving alarms or having observed symptoms. To locate the root cause accurately and timely in a large communication system is challenging because a single fault can often result in a large number of alarms, and multiple faults can occur concurrently. In this paper, we present a new fault localization method using a machine-learning approach. We propose to use logistic regression to study the correlation among network events based on end-to-end measurements. Then based on the regression model, we develop fault hypothesis that best explains the observed symptoms. Unlike previous work, the machine-learning algorithm requires neither the knowledge of dependencies among network events, nor the probabilities of faults, nor the conditional probabilities of fault propagation as input. The "low requirement" feature makes it suitable for large complex networks where accurate dependencies and prior probabilities are difficult to obtain. We then evaluate the performance of the learning algorithm with respect to the accuracy of fault hypothesis and the concentration property. Experimental results and theoretical analysis both show satisfactory performance.**

***Index Terms*—Complex networks, computer network reliability, fault diagnosis, fault location, logistic regression, machine learning.**

I. Introduction

IN A complex network with hundreds or thousands of software and hardware components, many of them are prone to failures. A failure of an important component, if not taken care of in a timely manner, can often lead to fault propagation in large areas and cause service disruption to many customers. It is, therefore, desirable that the network is equipped with fault management mechanisms in order to reduce network maintenance costs and improve service availability and reliability.

In the context of fault management, two important terms are frequently used: fault and symptom [1]–[3]. Faults are network events that cause other events. Faults are usually spontaneous and not caused by other events. Many faults may not be directly observable, but a seemingly invisible fault may manifest itself through failures at other observable locations. While faults are the root cause of failures, symptoms are external manifestation of failures.

Manuscript received August 18, 2015; revised October 11, 2015; accepted November 11, 2015. Date of publication November 24, 2015; date of current version September 08, 2016. The work of M. X. Cheng was supported by the National Science Foundation (NSF) under Grant ECCS-1307458, Grant CNS-1537538, Grant CNS-1545063, and Grant CMMI-1551448. The work of W. B. Wu was supported by the NSF under Grant DMS-1405410.

M. X. Cheng is with the Department of Computer Science, Missouri University of Science and Technology, Rolla, MO 65401 USA (e-mail: chengm@mst.edu).

W. B. Wu is with the Department of Statistics, University of Chicago, Chicago, IL 60637 USA (e-mail: wbwu@galton.uchicago.edu).

Digital Object Identifier 10.1109/JIOT.2015.2503270

Fault localization is the process of identifying the possible root cause(s) from the observed symptoms. Many fault localization techniques have been proposed and they vary from each other from the input/output interface to the algorithmic core of the procedure. According to the survey [4], there are close to a dozen of different fault localization techniques [1]–[3], [5]– [7]. Despite so much research effort, fault localization in complex networks remains an open research question. It is challenging mainly because the difficulty of fault localization increases with the size and complexity of the network—too much uncertainty, and too many unknown variables. In addition, fault localization techniques face the challenge of incomplete and incorrect input, which is inherited from its precursor, fault detection. Fault detection is the process of capturing symptoms provided in the form of events, and is carried out before fault localization. During the detection phase, there are false positives and false negatives, and therefore spurious symptoms and missing symptoms are possible.

In this paper, we aim to address these challenges and advance research in fault localization. In particular, we provide a machine-learning algorithm that uses whatever input data are given and makes the best of it. The algorithm is capable of handling difficult cases where there are loss of symptoms and spurious symptoms while the learner does not have the knowledge of fault probabilities of components, nor the conditional probabilities for fault propagation.

The proposed work focuses on the algorithmic aspect of fault localization in complex networks. The complete architecture of the network fault management system and other related activities in fault management are beyond the scope of this paper. This paper is organized as follows. In Section II, we briefly survey the previous related work; in Section III, we present the main idea of the machine-learning approach for fault localization; in Section IV, we provide analytical results for the error bound and the concentration property of the learning algorithm; in Section V, we show the application of the machine-learning algorithm on example networks and demonstrate its performance. Section VI concludes this paper and points out future research directions.

II. Related Work

Fault localization is the central piece in fault diagnosis. The literature has seen a variety of fault localization techniques [4]. The main stream of work either uses artificial intelligence techniques or some techniques based on graph modeling of the network events.

Expert systems are the most widely used AI techniques for fault localization. However, the existing fault localization methods based on rule-based [8], [9] systems, case-based systems [10], and model-based systems [11]–[13], neural networks [14], [15], and decision trees [16], [17] are all different from the proposed machine-learning technique. Most expert systems use rule-based representation of their knowledge base, and others differ from it with respect to the expert knowledge they use. For instance, rule-based systems use surface knowledge and model-based systems use deep knowledge about network connectivity and operation. Despite so much effort from the AI domain, none of the existing work uses regression analysis to quantify the correlation among network events, and most of them do not perform well for complex network topologies. This is the main difference between the proposed machine-learning algorithm and the previous work.

The techniques based on the graph modeling of network events are also referred to graph-theoretic techniques. The fault localization process depends on a fault propagation model of the system, which represents the causal relationships between network events. The graph is also called causality graph, or dependence graph. The fault propagation model contains all the faults and symptoms with conditional probability of a symptom given that a fault has occurred. Based on this model, many algorithms have been developed. They usually map the observed symptoms to nodes of the fault propagation model, and try to find the fault(s) that can explain the set of observed symptoms. For instance, the codebook approach uses this graph model to develop a codebook and then maps the observed codeword to a specific fault [18], the Bayesian inference approach uses this model to infer the posteriori probability based on the prior probabilities and conditional probabilities [2], [5]. There are also other graph-theoretical approaches such as the set cover approach that tries to identify a small set of faults to "cover" the symptoms [19]. These algorithms all have some identified limitations that the proposed machine-learning algorithm can improve upon.

1) *Compared to Bayesian inference approach*: Although there are mature methodology for Bayesian inference, such as belief network [20], these techniques require accurate knowledge of abstract and physical dependencies, i.e., the dependencies among network events and among system components. The efficiency and accuracy of the fault localization algorithm are dependent on the accuracy of this knowledge. However, in practice, the dependence relationships and the conditional probabilities are hard to obtain. Even with this knowledge, the problem of reasoning faults on a general graph remains an NP-hard problem.
2) *Compared to the codebook approach*: The proposed algorithm does not rely on graph modeling, nor does it require the accurate knowledge of dependences and probabilities. Although the representation of the design matrix in the regression analysis looks similar to the codebook used in the codebook approach, our approach does not require a causality graph as input to generate the training data. The training data can be obtained by various means, which makes it easy to apply in practice.
3) *Compared to the set cover approach*: The set cover approach requires probability of failure as input which is often unknown in practice. Moreover, the set cover approach is only concerned with using minimum cost (based on a cost function) to match the observed symptoms, and there is no penalty for covering the nonsymptoms. As a consequence, the algorithm can output faults that cause more symptoms than what have been observed. In addition, the set cover problem is an NP-hard discrete optimization problem in its own right.

III. Machine-Learning Approach

A. Overview

Many techniques have been developed based on the graph modeling approaches. Some require the knowledge of the causality graph, or dependency graph, and some require the knowledge of the probability for a component to fail. For real-world complex networks, to obtain the full causality graph is a challenge itself. Moreover, there is an additional challenge to obtain the probability distribution. In practice, what we observed is only a portion of the entirety. In such a situation, a black-box approach that does not depend on the entirety of information is desired. This motivates us to try to tackle the problem using a different approach—the machine-learning approach.

In the machine-learning approach, a learning algorithm studies the correlation of network events from a limited set of data, called training data, and then uses the training data to develop a hypothesis about the relationships of network events. The training data are often obtained from an expert who knows the observation $\mathbf{X}$ and the corresponding outcome y. After the learner has developed the hypothesis, upon receiving a new data point that has not been seen before, the learner applies the hypothesis to predict the outcome.

In network fault localization, $\mathbf{X}$ is the set of symptoms and y is the fault. During the process of developing hypothesis and predicting new outcome, there is no need to know the prior probability distribution of y or the conditional probability of $p(\mathbf{X}|y)$. In fact, there is also no need to know the exact causality graph, which is very hard to obtain in a real communication system.

Without the causality graph, we will use the available faults and symptoms to build a table. This table is also called the design matrix. The algorithm does not require accurate causal relationship as input. In network applications, the fault and symptom data are obtained from historical data that have been recorded from previous network events and diagnosis reports.

The proposed approach has the following advantages compared to the three approaches mentioned above.

1) All three approaches deal with NP-hard problems. The proposed approach avoids solving NP-hard problems and instead provides a probability of erroneous prediction.
2) The proposed approach scales well with the network size. If we are to locate the faulty link(s), the number of predictions needed is linear—not exponential—to the number of links, and each prediction only needs to use

local information, which is demonstrated in Section V-C. Moreover, the training phase, which is the slower part of the process, can be done at offline time, and prediction is done online, which makes it suitable for real-time fault localization.

3) Some approaches can only deal with binary observations, i.e., whether a symptom has occurred. The proposed method can deal with real-valued observations, such as data throughput or end-to-end delay.
4) The proposed approach does not require prior probability distribution of faults, or any assumption about fault distribution.
5) The proposed approach can achieve high robustness against spurious symptoms and loss of symptoms with fewer observations (compared to the codebook approach), and can adaptively adjust the fitting model to achieve higher robustness should the sampling process appear to be error-prone.

In addition, we provide performance bound analysis for prediction error, which can be used as a guideline to improve the hypothesis.

B. Supervised Learning Via Logistic Regression

In the learning algorithm, instead of viewing faults as causal events of symptoms, we view symptoms as explanatory factors to explain faults. Whenever a fault occurs, a set of symptoms are observed. The coexistence of the set of symptoms and the fault gives us ideas about which symptoms are correlated to the fault. The learning algorithm will further study the relationship between the fault and the symptoms. Once this relationship has been learned, in a future situation when the fault is unknown, the observation of symptoms is used to predict the probability of the fault. The fault localization procedure includes an estimation phase and a prediction phase.

1) Estimation: Without the knowledge of the exact causal relationships among network events, regression analysis is an effective way to estimate the relationships among them. In particular, we are not interested in estimating the relationships among symptoms; we are interested in knowing which symptoms are related to the network fault, and exploring the exact form of the relationship.

In the context of regression analysis, a fault corresponding to a software or hardware failure is regarded as a response variable or an outcome variable; a symptom corresponding to an observed alarm is regarded as a predictor variable. Next, we will focus on estimating the relationship between a response variable and several predictor variables.

There are many techniques for carrying out regression analysis. In general, the form of correlation is represented as a function of predictor variables, from which we can learn which predictor variables are related to a particular fault and how strong the correlations are. We propose to use logistic regression for fault localization.

Logistic regression, also called a logit model, is used to model dichotomous outcome variables. In this paper, the logit model defines the functional relationship between the alarm events and the actual probability of a component failure. Let $(x_1, x_2, \ldots, x_m)$ be the set of m predictor variables, y be the binary outcome, and p be the probability of y being 1. The logit model defines the log odds of the outcome as a linear combination of predictor variables

$$\log_e\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_m x_m = \boldsymbol{\beta}\cdot\mathbf{X} \tag{1}$$

where $\beta_1, \ldots, \beta_m$ are the regression coefficients indicating the relative effect of each particular explanatory variable on the outcome, and β_0 is the intercept. We use $\boldsymbol{\beta}$ to denote the vector $\{\beta_0, \beta_1, \beta_2, \ldots, \beta_m\}$ and use $\boldsymbol{X}$ to denote the vector $\{1, x_1, x_2, \ldots, x_m\}$.

The estimation process is to decide the values of $\beta_0, \beta_1, \ldots, \beta_m$ so that we can develop a hypothesis about the relationship between input variables $\mathbf{X}$ and the outcome y.

The logit function $\text{logit}(p) = \log_e\left(\frac{p}{1-p}\right)$ is a good choice for the fault localization problem since it can transform the probability p, a variable between 0 and 1, to any value from $-\infty$ to $+\infty$. This allows us to apply a generalized linear model to conduct linear regression for $\text{logit}(p)$ [21]–[23] and then use the inverse function of the logit function to recover the probability p later (see Section III-B2).

A good feature about this model is that the predictor variables are not limited to binary variables, unlike the codebook approach, in which each codeword is a binary string, or the set cover approach, in which whether a symptom is *covered* by a fault is a dichotomous outcome, or the Bayesian approach (see [5]), in which the symptoms are all categorical variables. In the logit model, all predictor variables are real-valued, therefore, the model can be used to study both the availability-related symptoms and performance-related symptoms.

2) Prediction: Prediction is the process of applying the hypothesis developed in the learning algorithm to compute the outcome from any predictor variables $\mathbf{X}$. At this step, the coefficient vector $\boldsymbol{\beta}$ is known from the estimation phase. For a given vector $\boldsymbol{X}$, it is easy to calculate the probability p of fault by using the inverse function of the logit function

$$\log_e\left(\frac{p}{1-p}\right) = \boldsymbol{\beta}\cdot\boldsymbol{X} \implies p = \frac{1}{1+e^{-\boldsymbol{\beta}\cdot\boldsymbol{X}}}. \tag{2}$$

If the probability is high, then we can conclude the fault has occurred. This is the classification problem for one particular fault. By applying the estimation and prediction process on each fault, we can get the probability of each fault. For F possible faults, there are F classification problems. Each classification problem has a separate training data set and generates a separate hypothesis. The dimension of the predictor variables $\mathbf{X}$ can be different for each hypothesis. Assume for the estimation of fault i, we apply hypothesis h_i, which takes predictor variables $\mathbf{X}^{(i)} \in R^{m_i}$ as input and maps to a probability between 0 and 1

$$h_i : (x_1, x_2, \ldots, x_{m_i}) \to p_i, 0 \le p_i \le 1 \qquad \forall i \in \text{Faults}.$$

If the symptoms are caused by the joint faults $f[i]$ and $f[j]$, then both p_i and p_j are close to or equal to 1. This is achieved

by sharing the observed symptoms in two separate set of training data. Suppose the joint faults $f[i]$ and $f[j]$ cause symptoms $\mathbf{X}^*$, then in the training data for $f[i]$, there is a data point $f^*[i]$ corresponding to $\mathbf{X}^*$, and in the training data for $f[j]$, there is also a data point $f^*[j]$ corresponding to $\mathbf{X}^*$. If the training data set shows that with input $\mathbf{X}^*$, 1 out of 1 case has $f^*[i] = 1$, then in the prediction phase, with input $\mathbf{X} = \mathbf{X}^*$, hypothesis h_i will yield $p_i = 1$, and, similarly, hypothesis h_j will yield $p_j = 1$.

The advantage of using this approach is that the complexity of estimation and prediction does not increase exponentially with the number of faults F. Since, we are going to predict the probability of each fault separately, there are always F classification problems to solve, which is linear to F. Moreover, joint faults do not increase the complexity of each classification problem either.

IV. Performance Bounds

To assess the performance of the learning algorithm for network fault localization, we would like to address the following questions.

1) Given the noise level of the data set, what is the expected estimation error for fault localization?
2) What is the concentration property of the estimation algorithm?

The first question is straightforward to answer given the error probabilities of observations, which only requires to calculate the expectation.

In the following, we will focus on the concentration property. The concentration property is an important performance measure for statistical learning algorithms. Taking the estimation result as a random variable, the concentration property tells us how far a particular estimation stands from the average estimation. The more concentrated around the average, the better.

The bounded differences method from McDiarmid provides an effective way to study the concentrate rate. McDiarmid's inequality provides an upper bound on the probability that the function of random variables deviates from its expected value [24]. It has been widely used in combinatorial applications and in learning theory [25], [26] to measure the concentration property of the learning algorithms.

We next show the concentration property of the network fault localization algorithm based on the McDiarmid's inequality [24].

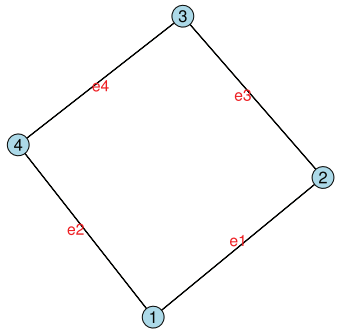
Theorem 1 (McDiarmid's Inequalityyn): Let $z_1, \ldots, z_m$ be independent random variables all taking values in the set $\mathcal{Z}$. Let $f : \mathcal{Z}^m \to \mathcal{R}$ be a function of $z_1, \ldots, z_m$ that satisfies $\forall i$

$$\sup_{z_1,\ldots,z_m,z_i' \in \mathcal{Z}} |f(z_1, \ldots, z_i, \ldots, z_m) - f(z_1, \ldots, z_i', \ldots, z_m)| \leq \ell_i$$

then for all $\epsilon > 0$

$$\mathbb{P}(f - \mathbb{E}f \geq \epsilon) \leq \exp\left(\frac{-2\epsilon^2}{\sum_{i=1}^m \ell_i^2}\right).$$

In the fault localization application, let m be the number of variables used in the estimation algorithm, which is the number of observations used in fault localization, the function f be the error probability of estimating the failure of a particular component, input $z_1, \ldots, z_m$ be the indicator variables of erroneous observations on the monitored event. Therefore, $z_i = 1$ if the ith observation is wrong (either a spurious symptom or a loss of symptom), otherwise $z_i = 0$. Since the probability of having a spurious symptom or a loss of symptom is random and is independent of others, it satisfies the condition of McDiarmid's inequality that $z_1, \ldots, z_m$ be independent random variable.



(a)

```
> print_table
   e1 e2 e3 e4 C12 C13 C14 C23 C24 C34 D12 D13 D14 D23 D24 D34
1   1  0  0  0   0   0   0   0   0   0   3   2   1   1   2   1
2   0  1  0  0   0   0   0   0   0   0   1   2   3   1   2   1
3   0  0  1  0   0   0   0   0   0   0   1   2   1   3   2   1
4   0  0  0  1   0   0   0   0   0   0   1   2   1   1   2   3
5   1  1  0  0   1   1   1   0   0   0 100 100 100   1   2   1
6   1  0  1  0   1   0   0   1   1   0 100   2   1 100 100   1
7   1  0  0  1   1   1   0   0   1   1 100 100   1   1 100 100
8   0  1  1  0   0   1   1   1   1   0   1 100 100 100 100   1
9   0  1  0  1   0   0   1   0   1   1   1   2 100   1 100 100
10  0  0  1  1   0   1   0   1   0   1   1 100   1 100   2 100
11  0  0  0  0   0   0   0   0   0   0   1   2   1   1   2   1
>
```

(b)

Fig. 1. (a) Network of four nodes and (b) design matrix, where Cij indicates end-to-end connectivity between node i and node j, Dij indicates end-to-end path length in hops. "100" represents a disconnected path. Columns $e1, \ldots, e4$ are classification results, where a "1" entry indicates the link is down.

Using the example network in Fig. 1, we show the error bounds of estimation. We are interested in the estimation error caused by erroneous input. Since the network is symmetric in topology, we use the estimation of one link $e1$ as an example. There are 12 observations, so $m = 12$. Let $z_1, \ldots, z_m = 0$, and $z_i' = 1$, then ℓ_i is the error probability for estimation when the ith observation (input) is wrong. The test results show that $\ell_1 = 1/r$, where r is the number of rows in the training data set, $\ell_7 = 0.5$, and $\ell_i = 0$ for $i \neq 1, 7$. The complete training data set includes 16 data points, so $r = 16$.

The numerical results for performance bounds are presented in the following.

A. Expected Estimation Error

Let $p_i = \mathbb{P}(z_i = 1)$ be the probability of the ith observation being wrong. Let $p_i = 0.1$ for $i = 1, \ldots, 6$, and $p_i = 0.01$ for $i = 7, \ldots, 12$. The expected estimation error is given by

$$\mathbb{E}f = \sum_i p_i \ell_i = 0.01125.$$

TABLE I
UPPER BOUND FOR $\mathbb{P}(f - \mathbb{E}f \geq \epsilon)$

ϵ	U
0.05	0.9805003
0.1	0.9242532
0.2	0.7297327
0.3	0.4921739
0.4	0.2835668
0.5	0.1395642

B. Concentration Property

Let U be the upper bound from McDiarmid's inequality, so $\mathbb{P}(f - \mathbb{E}f \geq \epsilon) \leq U$, then we have the following results.

It is observed from Table I that as the value of ϵ increases, the probability $\mathbb{P}(f - \mathbb{E}f \geq \epsilon)$ decreases sharply. If there is one erroneous input, the learning algorithm has less than 14% probability to yield a result that differs from the expectation 0.01125 by 0.5. Since the expectation is close to zero, the chance to have high error rate under one erroneous input is small. If the inputs are all correct, the estimation error is zero. Moreover, the loopy network we used here represents the most difficult case for the learning algorithm since it has a loop of the network size. Tree-shaped networks with no loops all have smaller estimation error and better concentration property.

V. EXAMPLES: FAULT LOCALIZATION THROUGH END-TO-END SERVICE DIAGNOSIS

To identify faulty elements through end-to-end measurements is a commonly used approach due to the availability of data at the end systems [27]–[29]. An end-to-end connectivity service model was proposed in [20] for loop-free networks. In this section, we first use the same service model as in [20] to infer faulty links, then we present our results on complex networks that have loops and tightly knit groups.

A. Complete Data, Perfect Learner

If the training data include all possible input points, the learned solution $h(\mathbf{X})$ is the ground truth plus some random noise. The only source of prediction error comes from the noise in the training data or the test data. Without noise, we will show that it has zero training error and zero test error.

We first look at a case without noise. In Fig. 2, there are nine nodes and eight links. The design matrix has $\binom{9}{2} = 36$ observations and $2^8 = 256$ data points. This includes all possible input data points, ranging from no faulty link to any number of faulty links. Each observation corresponds to the connectivity service disruption between a pair of end hosts.

If the data set has noise, the prediction error depends on the locations that are affected by noise. Noise can appear in observations, for instance, there will be loss of symptoms, or spurious symptoms; it can also appear in classification results, for instance, mistakenly labeling a data point as being faulty whereas the given input is for not being faulty, or vice versa.

The test results show that when an observation is corrupted due to noise, unless it is a critical observation, the solution is fairly robust to noise. In the nine-node network, for the

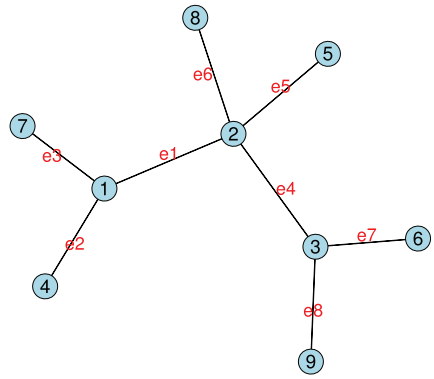


Fig. 2. Network of nine nodes.

Number of faulty links	Prediction accuracy (w/o noise)	Prediction accuracy (test data noise, I)	Prediction accuracy (test data noise, II)
0	100%	0%	100%
1	100%	0%	100%
2	100%	0%	100%
3	100%	0%	100%
4	100%	0%	100%
5	100%	0%	100%
6	100%	0%	100%
7	100%	0%	100%
8	100%	0%	100%

Fig. 3. Summary of prediction results for the network in Fig. 2.

prediction of whether link e1 between nodes 1 and 2 is faulty, the critical observation is the end-to-end connectivity between nodes 1 and 2. If link e1 is down, many pairs will lose end-to-end connectivity: {(1–2), (1–3), (1–5), (1–6), (1–8), (1–9), (2–4), (2–7), (3–4), (3–7), (4–5), (4–6), (4–8), (4–9), (5–7), (6–7), (7–8), (7–9)}. Assume losing connectivity is a symptom, then input "1" indicates the connection is lost and "0" indicates there is connection. Among all the observations, apparently the end-to-end connectivity between nodes 1 and 2 is the most important observation to predict the status of link e1. If this observation is corrupted, either it is "0" flipped to "1" representing a spurious symptom, or it is "1" flipped to "0" representing a loss of symptom, the prediction of link e1 has 100% prediction error. However, if other observations are corrupted, such as the end-to-end connectivity between nodes 1 and 5, the learning algorithm has zero prediction error.

The test results are summarized in Fig. 3. The training data are noise-free. When the test data are also noise-free, the test result is 100% accurate. When the test data have noise at the key location (column I), the result is 100% wrong; but when the data corruption is at a nonkey location, the result is still 100% correct (column II). In order to improve the performance when the test data have noise at the key location, simply adding redundant columns corresponding to this key observation will significantly increase the robustness, without adding redundancy for other observations. For instance, repeating three times in the training data for the column at the key location can tolerate one-bit data corruption at these columns. By using separate training data for each fault prediction, the design matrix size only increases by two columns, much smaller than the redundancy needed by the coding approach, which uses one

codebook to identify all possible faults. The advantage of the machine-learning approach to the codebook approach can be easily shown in this example: if the codebook approach is used, it does not matter which bit of the codeword is corrupted. In terms of hamming distance, the contribution from each bit in the codeword is the same, even though it is not at a critical position.

The test results also give much insight about the critical measures in fault diagnosis. The critical measures for predicting a link's status are the measurements from the two end points of the link. Errors in these measurements have higher impact on the outcome than in others. For links in a loop, such as link $e1$ in Fig. 1, the errors on connectivity information and delay both cause errors in predicting. For tree-type links, such as link $e1$ in Fig. 2, errors on the measurements of the two end points can completely invert the outcome of predicting.

Classification noise is another source for prediction error. The good news is that the prediction error is confined to that one data point only. Other data points without classification error can still have zero prediction error.

B. Incomplete Data, Imperfect Learner

Sometimes due to the unavailability of data, the training data set is very limited. It happens if there exist spontaneous multiple faults but the symptoms have never been recorded before. Since the learning algorithm has never seen the symptoms of multiple concurrent faults, it may not learn well enough to estimate the coefficients $\boldsymbol{\beta}$ accurately.

For the network in Fig. 2, if the training data only contain data points for singleton faults but the test data result from the combination of multiple faults, the prediction error becomes significant. For instance, link $e4$ is the link between nodes 2 and 3, and link $e7$ is the link between nodes 3 and 6. When links $e4$ and $e7$ both are faulty, the algorithm can only locate link $e4$, but not link $e7$. The reason is that the symptoms caused by disconnecting link $e4$ alone dominate the symptoms resulting from joint faults of links $e4$ and $e7$. In fact, the symptoms for joint faults of $e4$ and $e7$ have only two positions different from that of link $e4$ alone, but are very different from that of link $e7$ alone. In this case, the algorithm fails to locate link $e7$. We say link $e4$ dominates link $e7$, and similarly, link $e4$ also dominates link $e8$. This dominating relation continues if the test data are symptoms resulted from three simultaneous faults. However, if we extend the training data to include the symptoms of two concurrent faults, experiments show that the algorithm has zero prediction error for not only two concurrent faults but also for three concurrent faults. It works for all test cases with 100% accuracy. For instance, for the joint faults of $\{e4, e7, e8\}$, the algorithm can locate $e4$ and $e7$ since $\{e4, e7\}$ is a data point in the training set, and similarly it can also locate $e4$ and $e8$, then the union of the two sets $\{e4, e7\} \bigcup \{e4, e8\}$ yields $\{e4, e7, e8\}$.

The test results are summarized in Fig. 4. When the training data have eight data points including the observations for only one faulty link at a time, the prediction error is high in some cases; when the training data has 36 data points including the observations for one faulty link and two faulty links at a time, the prediction error falls back to zero, even if the test data contain higher than two degree of interaction between faults.

Training set data points	Prediction accuracy		
	1faulty link	2 faulty links	3 faulty links
8	100%	21 cases 100% correct 7 cases 50% correct (1,4), (1,5),(1,6),(1,7),(1,8): 1 dominates {4,5,6,7,8} (4,7),(4,8): 4 dominates {7,8}	 1 dominate {4,5,6,7,8} 4 dominates {7,8}
36	100%	100%	100%

Fig. 4. Summary of prediction results for the network in Fig. 2 when the training set is incomplete.

Training set data points	Test set data points	Prediction accuracy	
		Training set	Test set
25	11	All 25 cases 100%	9 cases 100% 2 cases 50% (4,7),(4,8): located 4 4 dominates {7, 8}
64	28	All 64 cases 100%	All 28 cases 100%

Fig. 5. Summary of prediction results for the network in Fig. 2 using the 70–30 rule.

A common practice is to divide the available data into training data and test data following a 70–30 rule: 70% data are used as training data, and 30% data are used as test data. Following this rule, we did two experiments: 1) the 36 data points are split into a training set of 25 data points and a test set of 11 data points, and the training set includes all singleton faults and partial data from two faults and 2) the 92 data points are split into a training set of 64 data points and a test set of 28 data points, and the training set includes all singleton faults, all data from two faults and partial data from three faults. The result is shown in Fig. 5. The prediction accuracy is significantly improved. Only two cases have achieved 50% accuracy: for joint faults of $\{e4, e7\}$, the algorithm only found $e4$; for joint faults of $\{e4, e8\}$, the algorithm only found $e4$. All other cases have achieved 100% accuracy.

C. Large-Scale Complex Networks

While small networks allow us to present the evaluation results with great detail, large-scale networks will require much larger space to present the same level of detail. For large networks, we randomly select a few faulty links and present the summary results. Fig. 6 shows a network with 50 nodes. The six faulty links are randomly selected and highlighted in red color. The network is loosely hierarchical with cycles and densely connected subgraphs (cliques), which is similar to the backbone structure of the Internet. We choose to evaluate a graph with cycles and cliques because such structures represent the highest level of challenge for fault localization algorithms.

1) High-Dimensional Inference: If we blindly include all available observations without selection for the estimation of any link, the problem becomes a high-dimensional inference problem. For instance, if we include all end-to-end connectivity data, the dimension of $\mathbf{X}$ will be $\binom{50}{2} = 1225$ in the logistic

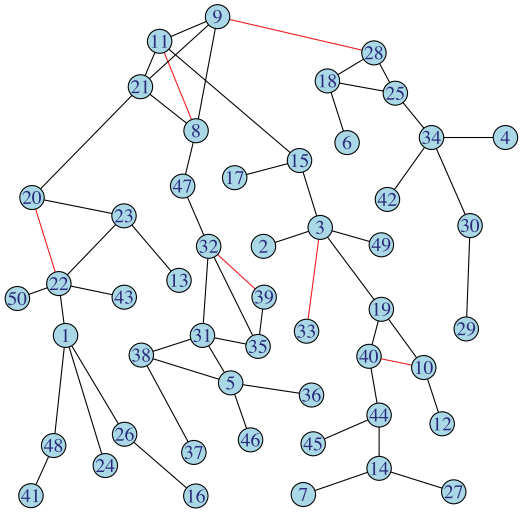


Fig. 6. Network with loops and cliques.

regression analysis. The result we achieved is summarized as follows.

For tree-like links such as links 9–28 and 3–33, we can identify the faulty link with 100% accuracy, which means when using the observation of link 9–28 being broken, it can predict that link 9–28 has probability 1 to be faulty while others have probability 0. However, for links involved in a loop or multiple loops, such as links 8–11, 20–22, 32–39, and 10–40, when the link is faulty, the prediction result shows it only has probability $1/r$ to be faulty, where r is the number of rows in the training set that has the same observation. This is due to the fact that when the link is broken, the end-to-end connectivity is still preserved by using alternative routes.

In order to improve the prediction accuracy, we include the end-to-end delay in the observation. This will increase the dimension of $\mathbf{X}$ to $\binom{50}{2} \times 2 = 2450$. With such a high dimension, we can finally achieve 100% accuracy on all links, including the tree-like links and the links in loops. Moreover, if the training data include observations for all single link faults, the test data with two or more combined link faults can be predicted with 100% accuracy.

2) Simplistic Low-Dimensional Inference: High-dimensional inference significantly increases the computation time, which is undesired in real-time applications. In addition, from a practical point of view, to obtain observations for all pairs of nodes requires global information, which becomes an issue in distributed implementation. To reduce the dimension and to make it easy for distributed implementation, we choose only the most relevant explanatory factors. For instance, to predict whether a link is up or down, the most explanatory factors are the connectivity and delay information between the two end points of the link. Using only the two factors, we can achieve the following: when the training set includes the observation for a particular fault, the prediction result is 100% accurate, for both tree-like links and links in a loop. It does not matter if the training set includes observations for other faulty links. It can also predict two or more combined link faults with 100% accuracy. However, when the training set does not include it, the prediction result shows $\mathbb{P}(\text{faulty}) = 0$ where it should be 1. This makes sense because there is nothing in the training set to infer from.

3) Robust Low-Dimensional Inference: The above observation indicates that some degree of redundancy in the parameters is necessary. Instead of using only two factors, we include the end-to-end connectivity and delay for all pair of nodes in a link's neighborhood. For instance, to estimate link 9–28, we consider nodes $N_{9,28} = \{9, 8, 11, 21, 28, 18, 25\}$ and use the end-to-end connectivity and delay between all pair of nodes in this set. By expanding the dimension, we achieved the following. 1) All prediction are 100% accurate. Even if the training set does not include a row corresponding to a link being faulty, the prediction on this link is still 100% correct. 2) The prediction of two or more links being faulty at the same time is also 100% correct. 3) Sometimes we fail to obtain the measurement for a particular link. For instance, we may not have data for end-to-end connectivity or delay between nodes x–y, which will require the elimination of the two factors in regression analysis. Even without the two key factors, with the neighborhood information, the prediction of a link is still improved. For instance, if we remove the two key factors for link 9–28, it still can predict this link's status as well as others with 100% accuracy, for both singleton faults and combined faults.

In summary, the simplistic low-dimensional inference method in Section V-C2 works well only if the two key measurements are available and the training set includes the observation for the link being faulty. Expanding the dimension by including some nodes in the neighborhood effectively improves the algorithm's robustness. It is also worth noting that the expanded dimension is still in much smaller magnitude compared to the *include-all* method in Section V-C1. With seven nodes, it only has 42 factors, much smaller than the 2450 factors used in high-dimensional inference.

VI. Conclusion and Future Work

The proposed fault localization approach uses logistic regression to study the correlation between the root cause and the observed end-to-end service degradation. The end-to-end service degradation can be manifested in both availability-related symptoms and performance-related symptoms. The approach is shown to be effective in identifying faults and is also proved to have good concentration property.

The current work focuses on fault localization, which is solved by predicting dichotomous outcome variables using binomial logit models. In future work, one immediate extension is to consider multiclass classification problems by using multinomial logit models. For instance, we can estimate "Is the router status normal, overloaded, or being attacked?" Another future direction is to consider nondichotomous outcomes, in which the prediction is made on a continuous variable y. For instance, instead of asking "is link i faulty?" we ask "what is the data rate of a core link?" "what is the loss rate of a wireless link?" or "what is the average queue length at a router?", etc. This will be general network state inference beyond fault localization.

REFERENCES

[1] I. Katzela and M. Schwartz, "Schemes for fault identification in communication networks," *IEEE/ACM Trans. Netw.*, vol. 3, no. 6, pp. 753–764, Dec. 1995.

[2] W. Fischer, G. Xie, and J. Young, "Cross-domain fault localization: A case for a graph digest approach," in *Proc. IEEE Internet Netw. Manage. Workshop (INM'08)*, Oct. 2008, pp. 1–6.

[3] G. Reali and L. Monacelli, "Definition and performance evaluation of a fault localization technique for an NGN IMS network," *IEEE Trans. Netw. Serv. Manage.*, vol. 6, no. 2, pp. 122–136, Jun. 2009.

[4] M. Steinder and A. S. Sethi, "A survey of fault localization techniques in computer networks," *Sci. Comput. Program.*, vol. 53, no. 2, pp. 165–194, 2004.

[5] M. Steinder and A. Sethi, "Increasing robustness of fault localization through analysis of lost, spurious, and positive symptoms," in *Proc. 21st Annu. Joint Conf. IEEE Comput. Commun. Soc. (INFOCOM)*, 2002, vol. 1, pp. 322–331.

[6] R. R. Kompella, J. Yates, A. Greenberg, and A. C. Snoeren, "IP fault localization via risk modeling," in *Proc. 2nd Conf. Symp. Netw. Syst. Des. Implement.*, 2005, pp. 57–70.

[7] S. Kandula, D. Katabi, and J.-P. Vasseur, "Shrink: A tool for failure diagnosis in IP networks," in *Proc. ACM SIGCOMM Workshop Mining Netw. Data*, 2005, pp. 173–178.

[8] G. Liu, A. Mok, and E. J. Yang, "Composite events for network event correlation," in *Proc. 6th IFIP/IEEE Int. Symp. Integr. Netw. Manage. Distrib. Manage. Netw. Millennium*, 1999, pp. 247–260.

[9] M. Klemettinen, H. Mannila, and H. Toivonen, "Rule discovery in telecommunication alarm data," *J. Netw. Syst. Manage.*, vol. 7, no. 4, pp. 395–423, 1999.

[10] L. Lewis, "A case-based reasoning approach to the management of faults in communications networks," in *Proc. 9th Conf. Artif. Intell. Appl.*, Mar. 1993, pp. 114–120.

[11] S. Brugnoni, G. Bruno, R. Manione, E. Montariolo, E. Paschetta, and L. Sisto, "An expert system for real time fault diagnosis of the Italian telecommunications network," in *Integrated Network Management*, H.-G. Hegering and Y. Yemini, Eds. Amsterdam, The Netherlands: North Holland, 1993, vol. C-12, pp. 617–628.

[12] S. Fontanini, J. Wainer, V. Bernal, and S. Maragon, "Model based diagnosis in LANs," in *Proc. IEEE Workshop IP Oper. Manage.*, 2002, pp. 121–125.

[13] L. N. D. Barros and M. Lemos, "Model based diagnosis for network communication faults," in *Proc. Int. Workshop Artif. Intell. Distrib. Inf. Netw. (AIDIN'99)*, 1999, pp. 57–62.

[14] R. Gardner and D. Harle, "Pattern discovery and specification techniques for alarm correlation," in *Proc. IEEE/IFIP Netw. Oper. Manage. Symp.*, Feb. 1998, vol. 3, pp. 713–722.

[15] H. Wietgrefe, "Investigation and practical assessment of alarm correlation methods for the use in GSM access networks," in *Proc. IEEE/IFIP Netw. Oper. Manage. Symp.*, 2002, pp. 391–403.

[16] B. Wang, W. Wei, H. Dinh, W. Zeng, and K. R. Pattipati, "Fault localization using passive end-to-end measurements and sequential testing for wireless sensor networks," *IEEE Trans. Mobile Comput.*, vol. 11, no. 3, pp. 439–452, Mar. 2012.

[17] G. D. Rodosek and T. Kaiser, "Intelligent assistant: User-guided fault localization," in *Proc. 9th Int. Workshop Distrib. Syst. Oper. Manage.*, 1998, pp. 119–129.

[18] S. Kliger, S. Yemini, Y. Yemini, D. Ohsie, and S. Stolfo, "A coding approach to event correlation," in *Proc. 4th Int. Symp. Integr. Netw. Manage. IV*, 1995, pp. 266–277.

[19] A. Bouloutas, S. Calo, and A. Finkel, "Alarm correlation and fault identification in communication networks," *IEEE Trans. Commun.*, vol. 42, no. 234, pp. 523–533, Feb. 1994.

[20] M. Steinder and A. Sethi, "Probabilistic fault localization in communication systems using belief networks," *IEEE/ACM Trans. Netw.*, vol. 12, no. 5, pp. 809–822, Oct. 2004.

[21] P. McCullagh and J. A. Nelder, *Generalized Linear Models*. London, U.K.: Chapman & Hall, 1990.

[22] J. S. Long, *Regression Models for Categorical and Limited Dependent Variables*. Newbury Park, CA, USA: Sage, 1997.

[23] A. J. Dobson and A. G. Barnett, *An Introduction to Generalized Linear Models*. Boston, MA, USA: Chapman & Hall, 2008.

[24] C. McDiarmid, "On the method of bounded differences," *Surveys in Combinatorics*, J. Siemons Ed. Cambridge, U.K.: Cambridge Univ. Press, 1989, pp. 148–188.

[25] O. Bousquet and A. Elisseeff, "Stabilty and generalization," *J. Mach. Learn. Res.*, vol. 2, pp. 499–526, 2002.

[26] P. L. Bartlett and S. Mendelson, "Rademacher and Gaussian complexities: Risk bounds and structural results," *J. Mach. Learn. Res.*, vol. 3, pp. 463–482, 2002.

[27] P. Lee, V. Misra, and D. Rubenstein, "Toward optimal network fault correction via end-to-end inference," in *Proc. 26th IEEE Int. Conf. Comput. Commun.*, May 2007, pp. 1343–1351.

[28] M. Fraiwan and G. Manimaran, "Localization of IP links faults using overlay measurements," in *Proc. IEEE Int. Conf. Commun. (ICC'08)*, May 2008, pp. 5629–5633.

[29] A. Hanemann and P. Marcu, "Algorithm design and application of service-oriented event correlation," in *Proc. 3rd IEEE/IFIP Int. Workshop Business-Driven IT Manage. (BDIM'08)*, Apr. 2008, pp. 61–70.

Maggie X. Cheng received the Ph.D. degree in computer science from the University of Minnesota, Minneapolis, MN, USA, in 2003.

She is currently an Associate Professor with the Computer Science Department, Missouri University of Science and Technology, Rolla, MO, USA. Her research interests include data analytics for computer networks and cyber-physical systems, network anomaly detection, and fault diagnosis.

Wei Biao Wu received the Ph.D. degree in statistics from the University of Michigan, Ann Arbor, MI, USA, in 2001.

He is currently a Professor of Statistics with the University of Chicago, Chicago, IL, USA. His research interests include probability theory, statistics, econometrics, and statistical inference for high-dimensional data.