Metric Ranking of Invariant Networks with Belief Propagation

Changxia Tao[1], Yong Ge[2], Qinbao Song[1], Yuan Ge[3], Olufemi A. Omitaomu[4]
[1]Xi'an JiaoTong University, {taoxixi413@stu, qbsong@mail}.xjtu.edu.cn
[2]The University of North Carolina at Charlotte, yong.ge@uncc.edu
[3]Anhui Polytechnic University, ygetoby@mail.ustc.edu.cn
[4]Oak Ridge National Laboratory, omitaomuoa@ornl.gov

***Abstract*—The management of large-scale distributed information systems relies on the effective use and modeling of monitoring data collected at various points in the distributed information systems. A promising approach is to discover invariant relationships among the monitoring data and generate invariant networks, where a node is a monitoring data source (metric) and a link indicates an invariant relationship between two monitoring data. Such an invariant network representation can help system experts to localize and diagnose the system faults by examining those broken invariant relationships and their related metrics, because system faults usually propagate among the monitoring data and eventually lead to some broken invariant relationships. However, at one time, there are usually a lot of broken links (invariant relationships) within an invariant network. Without proper guidance, it is difficult for system experts to manually inspect this large number of broken links. Thus, a critical challenge is how to effectively and efficiently rank metrics (nodes) of invariant networks according to the anomaly levels of metrics. The ranked list of metrics will provide system experts with useful guidance for them to localize and diagnose the system faults. To this end, we propose to model the nodes and the broken links as a Markov Random Field (MRF), and develop an iteration algorithm to infer the anomaly of each node based on belief propagation (BP). Finally, we validate the proposed algorithm on both real-world and synthetic data sets to illustrate its effectiveness.**

Keywords*— *Invariant; ARX Model; Invariant Networks; Belief Propogation

I. Introduction

Recent advances in information infrastructure have enabled the development of networked information systems. These large-scale information systems usually consist of thousands of components, such as servers, networking devices, and storage equipment. The connectivity of these devices and hence the complexity of the information systems they are embedded in correlates well with their functional essentiality. Also, the dynamics and heterogeneity of such information systems introduces another dimension of complexity. Therefore, it has been a great challenge to maintain and manage these large-scale, dynamic and complex information systems.

In previous work [1], [2], [3], Jiang et al. have proposed a System Invariant Analysis Technique (SIAT) to model the system dynamics and detect system faults. They used the AutoRegressive models with eXogenous inputs (ARX) [1], [4] to model the relationships between each pair of monitoring data, such as the number of SQL Queries and the average memory usage. Specifically, the ARX model between two monitoring data x and y is $y(t)+a_1y(t-1)+\cdots+a_ny(t-n) = b_0x(t-k)+\cdots+b_mx(t-k-m)$. If such a relationship between x and y holds all the time when there is no fault, they regarded it as an invariant of the underlying information systems. For example, one relationship based on the ARX model may be $y_{ejb}(t) = 0.08y_{ejb}(t-1)-0.39x_{jvm}(t)$, where y_{ejb} and x_{jvm} represent the intensity of 'EJB created' and 'JVM processing time'. And this relationship (equation) is considered as an invariant if it holds all the time. Given all monitoring data of an information system, it is possible to generate an invariant network as shown in Figure 1, where each node represents a monitoring data (metric) and one edge between two nodes denotes an invariant (relationship) between these two metrics. A system fault often causes some monitoring data to change unusually and such change then causes some invariants to be broken. At any future timestamp, we can get all broken links (invariants) in an invariant network. System experts can then follow the broken links and related metrics to narrow down possible problems and faults. In this paper, we propose to rank the metrics based on the anomaly level of each metric. The ranking of metrics can provide a guidance for system experts to follow the broken links and localize the fault more effectively.

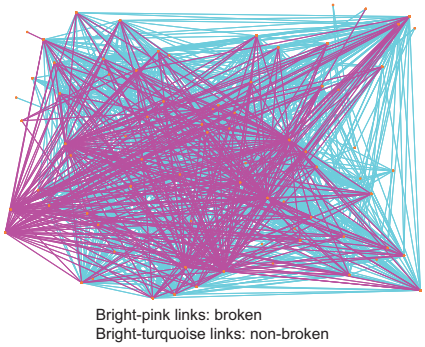

Fig. 1. An Example of Invariant Networks

Nonetheless, it is a nontrivial task to rank the metric anomaly in invariant networks. For example, in Figure 1, we show an example of invariant networks, where bright-pink links (lines) are broken and bright-turquoise ones are held at this timestamp. In the figure, we can observe the complexity of connectivity in invariant networks. Such complexity can vary among different invariant networks of different information systems. When we try to determine if one node is abnormal

or not in such complex invariant networks, we cannot only consider all associated links with this node, since the broken links among these associated links are also connected with other nodes. In other words, system faults can propagate from one node to another node, and thus all nodes associated with broken links are naturally tied together. As a result, it is not easy to decide which nodes (metrics) are abnormal and cause the broken links. Indeed, one fundamental challenge underlying this metric ranking problem is that one node (metric) with a certain degree of anomaly has a certain possibility to cause some of its related links to be broken. In this paper, we name this challenge as *uncertainty* for short. All *uncertainty* associated with each broken-link-related node together makes this metric ranking problem quite difficult.

To address the uncertainty challenge, in this paper, we propose to use Markov Random Fields (MRFs) to model the nodes and broken links. Each node can be in two states: normal and abnormal. With the modeling of MRFs, our goal is to infer the states of all nodes by considering all nodes and the connections among them together. To this end, we develop one algorithm based on the loopy belief propagation (LBP), which has been shown to be a very effective method for inferring the conditional marginal probability of MRFs. Finally, we examine the performances of the proposed algorithm on both real-world and synthetic data sets.

II. Preliminaries

A. Invariant Modeling and Extraction

In [1], [2], [3], Jiang et al. used AutoRegressive models with eXogenous inputs (ARX) [4] to learn the relationship between monitoring data. Following the notation in [2], at time t, we denote two monitoring data by $x(t)$ and $y(t)$. The ARX model describes the following relationship between them:

$$\begin{aligned} &y(t) + a_1 y(t-1) + \cdots + a_n y(t-n) \\ &= b_0 x(t-k) + \cdots + b_m x(t-k-m), \end{aligned} \tag{1}$$

where $[n, m, k]$ is the order of the model. a_i and b_j are the coefficient parameters. Let us denote

$$\varphi(t) = [-y(t-1), \cdots, -y(t-n), x(t-k), \cdots, x(t-k-m)]^T. \tag{2}$$

Then, Equation 1 can be rewritten as $y(t) = \varphi(t)^T\theta$. Assuming that we have observed $x(t)$ and $y(t)$ over a time interval $1 \le t \le N$, let us denote this observation by $O_N = \{x(1), y(1), \cdots, x(N), y(N)$. With a given θ and the observations, we can calculate the simulated $\hat{y}(t/\theta)$ according to Equation 1. Thus, we can get the estimation error as

$$E_N(\theta, O_N) = \frac{1}{N}\sum_{t=1}^{N}(y(t) - \hat{y}(t/\theta))^2 = \frac{1}{N}\sum_{t=1}^{N}(y(t) - \varphi(t)^T\theta)^2. \tag{3}$$

The Least Squares Method (LSM) can find the $\hat{\theta}$ that minimizes the estimation error $E_N(\theta, O_N)$. Then, the normalized fitness score $F(\theta)$ [2] is used to evaluate how well the learned model fits the real observations as

$$F(\theta) = 1 - \sqrt{\frac{\sum_{t=1}^{N}|y(t) - \hat{y}(t/\theta)|^2}{\sum_{t=1}^{N}|y(t) - \bar{y}|^2}}, \tag{4}$$

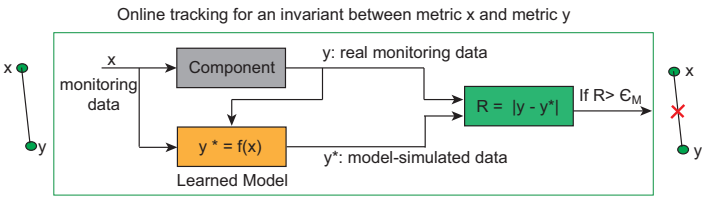

Fig. 2. An Invariant Online Tracking Example

where $\bar{y}$ is the mean of the real output $y(t)$. For one pair of metrics (monitoring data), if the highest fitness score among all model candidates is higher than a certain threshold τ, we treat the corresponding model with highest fitness score as an invariant.

B. Tracking of Invariant Networks

Given all monitoring data of an information system, invariant search is performed among all pairs of metrics with the observations of each metric. Then, a set of invariants can be obtained for this information system. If we represent each metric as a node and each invariant as a link between two metrics, we are able to generate a graph as shown in Figure 1, which is called an invariant network. At each timestamp t in the future, for each invariant we compare the real observation $y(t)$ and its simulated one $\hat{y}(t)$ calculated with $\hat{\theta}$ to get the absolute difference as

$$R_t = |y(t) - \hat{y}(t)|. \tag{5}$$

In the normal situation without fault, we should have the residual $R_t \le \epsilon_M$, where ϵ_M is a threshold of model error. If a system fault occurs inside the information system, it usually affects the metric relationship, and the invariants are likely to be violated. Thus we can observe such a fault in the real time by tracking whether the real output stays in the same trajectory as the invariant model expects, that is at time t, check whether $R_t \le \epsilon_M$. In Figure 2 we illustrate the invariant tracking for FDI, where the link between x and y is broken if $R > \epsilon_M$. Note that ϵ_M is different for each invariant. Actually, ϵ_M of each invariant is automatically decided with the residual distribution associated with each link. Specifically, we select a threshold $\epsilon_M = 1.1 \cdot arg_{\hat{R}}\{prob(|R(t)| < \hat{R}) = 0.995\}$, i.e., choose a value $\hat{R}$ that is larger than 99.5 percent of the observed residuals (after a long time period T) and the selected threshold is 1.1 times $\hat{R}$.

III. Problem Formulation

A. Metric Anomaly Ranking

By the online tracking of each invariant, the state (broken or not) of every invariant can be decided at a certain timestamp. After the states of all the links in an invariant network at a certain timestamp have been decided, the metric anomaly ranking problem can be formulated as follows.

Given an invariant network I with N nodes (metrics), denoted as $V_i, 1 \le i \le N$, and M links (invariants), denoted as $E_j, 1 \le j \le M$, there are a learned parameter $\hat{\theta}_j$, a set of residuals $\mathbb{R}_j$, and a fitness score F_j associated with each link E_j. Furthermore, we can calculate a threshold ϵ_M^j from

$\mathbb{R}_j$, where ϵ_M^j is used to check if invariant E_j is violated or not. Then, at a certain timestamp T, we can obtain an overall state $\mathbb{S}$ for these M links, which shows the states of all the invariants at T. The objective is to rank all metrics from most abnormal to least abnormal. Such metric anomaly ranking can provide a guidance for system experts to effectively and efficiently examine metrics and localize the system fault. Based on the above description and notations, we formally state the metric anomaly ranking problem as follows.

The Metric Anomaly Ranking Problem
Given: An invariant network I with N nodes $V_i\,(1 \leq i \leq N)$ and M links $E_j\,(1 \leq j \leq M)$, additional information $[\hat{\theta}_j, \mathbb{R}_j, F_j]$ associate with each link E_j, and the state of M links at a certain timestamp T, when the state of each link is either broken or not.
Objective: Ranking all N nodes from the most abnormal to the least abnormal

Note that the given invariant network I in the above Metric Anomaly Ranking Problem is a connected graph, which means there is a path between any two nodes. Given all metrics, it is possible that we may generate two or more completely-separated invariant networks with methods described in section II, but there will be no influence between the metric rankings on the separate invariant networks.

IV. Anomaly Ranking Algorithm

After the states of all the links in an invariant network at a certain timestamp have been decided, we then model the nodes and the broken links as a Markov Random Field (MRF) and employ the Loopy Belief Propagation (LBP) method to infer whether a node (i.e., a metric) being abnormal or not.

A. Modeling Invariant Network with MRF

Markov Random Fields (MRFs) are a class of probabilistic graphical models particularly suitable for solving inference problems with uncertainty in observed data. There are two types of nodes in MRF: observed nodes and hidden nodes. Observed nodes correspond to values that are actually observed in the data. For each observed node, there is a hidden node which represents the true state underlying the observed value. The state of a hidden node depends on the value of its corresponding observed node as well as the states of its neighboring hidden nodes. Such dependency is captured via an edge compatibility function $\Psi(\lambda, \lambda')$ and a node compatibility function $\Phi(\lambda, \omega)$. $\Psi(\lambda, \lambda')$ gives the probability of a hidden node being in state λ' given that it has a neighboring hidden node in state λ. $\Phi(\lambda, \omega)$ gives the probability of a node being in state λ given that its corresponding observation was ω. There is a belief vector b_i associated with each hidden node i, and $|b_i|$ denotes the number of states which the hidden node can be in. $b_i(\lambda)$ is the probability of node i being in state λ (i.e., the belief of node i in state λ).

The invariant network can naturally be modeled as a MRF. Specifically, to make the MRF adapted to our invariant network, we extract all nodes with at least one broken link and treat those nodes as hidden nodes of MRF. And all broken links are extracted as the links between those hidden nodes. For each hidden node, there are two possible states, which are abnormal and normal, denoted as S. Moreover, each hidden node is associated with an observed node, which corresponds to our initial (and possibly noisy) observation of its state in the data. Therefore, we may treat all broken links and all nodes connected to at least one of these broken links as a MRF.

To completely adapt the MRF to the invariant network, we need to instantiate the compatibility functions Ψ and Φ. When there is no prior or domain knowledge, we may choose Φ such that $\Phi(\lambda, \omega) = 1/|S|, \forall \lambda, \omega$. The edge compatibility function Ψ can be considered as a matrix (which is called Propagation Matrix) of dimension $|S| \times |S|$. Table I shows a sample instantiation. Entry (i, j) gives the conditional probability that the destination node is in state j given the source node is at state i. Such an instantiation is actually based on the following natural intuition: due to the fact that a broken link is caused by at least one of nodes it links, a normal node tends to link to an abnormal node but not to other normal nodes via a broken link. Thus we may set ε_0 to be a small value (e.g., 0.05) instead of zero because the numerical problems with multiplications. This reflects the fact that it is very unlikely that two nodes of a broken link are both normal. And since the possibility that two nodes of a broken link are abnormal simultaneously is also small, an abnormal node is more likely to connect with a normal node. Thus ε can be set as a value under 0.5. Ideally, we should "learn" the values of ε_0 and ε, as well as the form of the propagation matrix itself, if we had a large training set. However, in reality we may need to empirically specify both values instead.

TABLE I. An Example Propagation Matrix

	normal	abnormal
normal	ε_0	$1-\varepsilon_0$
abnormal	$1-\varepsilon$	ε

B. Ranking Algorithm based on Loopy BP

In the MRF, nodes influence each other and cyclic dependency exists between nodes. Traditional methods compute the probability of event occurrence through integration. But as the number of nodes becomes large, the relationship between nodes becomes complex. This would make it difficult to determine the states of nodes with traditional methods. Although Monte Carlo methods have been proposed, they could not fundamentally solve the problem. This difficulty had hindered the development of statistical inference until the raising of belief propagation which is a very effective method for inferring the conditional marginal probability. The main idea of BP is to localize and distribute the inference, i.e. transforming global integration to local message passing. BP takes a network of nodes as an input, each of which can be in a finite number of states. Some information about how the state of a node influences its neighbors is also known beforehand. Then the BP algorithm infers the posterior probability of states of all nodes in the network given the observation of the network nodes. Loopy belief propagation (LBP) is an extension of BP which is appropriate for the network with loops like MRF. The inference process can be summarized as follows.

The algorithm functions via iterative message passing between the different nodes in the network. Let m_{ij} denote the message vector that node i passes to node j, and its dimension is the same as the number of states each node can be in. In

this paper, the message m_{ij} is a vector with two dimensions because there are two states (i.e., normal and abnormal) for node of the invariant network. $m_{ij}(\lambda)$ represents node i's belief about that node j will be in the corresponding state λ. At every iteration, each node i computes its beliefs based on messages received from its neighbors, and uses the propagation matrix to transform its beliefs as messages to its neighbors.

In many cases, there is no extra information about the states of nodes except the structure of the network and the propagation matrix based on intuition of problem. In these cases, each node is initialized with an unbiased state. The belief about a state of a node is proportional to the product of all the messages coming into the node as:

$$b_i(\lambda) = k \prod_{j \in N(i)} m_{ji}(\lambda), \tag{6}$$

where k is a normalization term because the beliefs must sum to 1. And $N(i)$ denotes the set of its neighboring nodes. As can be seen from equation IV-B, there is no prior or local observation used for calculating the belief at a node.

The message m_{ij} from node i to node j is proportional to the product of all the messages from node i's neighboring nodes except node j:

$$m_{ij}(\lambda) = \sum_{\lambda'} \Psi(\lambda, \lambda') \prod_{n \in N(i) \setminus j} m_{ni}(\lambda'), \tag{7}$$

Usually the initial message of all nodes is set as $(1, \cdots, 1)$, which means a node believes that any of its neighboring nodes is in any of the possible states with equal probability. Then messages flow iteratively through the graph until the beliefs converge or a maximum number of iterations is reached. By the end of iteration, the state of each node is to reach a fixed point (equilibrium), that is states of all nodes are compatible with their neighbors as much as possible.

However, original LBP is known to produce poor models due to the lack of the prior information and local observation about the states of nodes. To this end, we introduce a pior to each hidden node of invariant networks. Also we define and combine the local observation at each node in our work. Specifically, based on the domain knowledge of information system faults and their influence pattern in invariant networks, we identify the following two important features to characterize the behavior of abnormal nodes and use them as the prior and the local observation for each node respectively.

Definition 1: Ratio of Broken links (RB). According to the cause of broken links we may have an intuition that a node is likely to be abnormal if most links of this node are broken. Along this line, we define the RB of a node V_i as:

$$RB_{V_i} = \frac{number\ of\ broken\ links\ of\ V_i}{number\ of\ all\ links\ of\ V_i}. \tag{8}$$

Generally speaking, an abnormal node should have a higher RB than that of a normal node.

Definition 2: Ratio of Unbroken links of BINNs (RUB). It is also possible that the broken links of a node are caused by its neighbors. Therefore, we need to consider the links related to the broken-invariant-neighboring-nodes (BINNs). The BINNs of a node are nodes each of which connects to this node with a broken link. If most links of BINNs of a node are not broken, this node is more likely to be abnormal. Specifically we define the RUB of a node as:

$$RUB_{V_i} = 1 - \frac{number\ of\ broken\ links\ related\ to\ BINNs}{number\ of\ all\ links\ related\ to\ BINNs}. \tag{9}$$

We use RB as the state prior for each node because we think that this is a more reliable indicator than RUB, which is used as the local observation of the state of each node. We believe that such a combination would yield the following benefits: (a) using a strong prior (i.e., RB) and a local observation (i.e., RUB) will help the belief propagation to converge to a more accurate solution with less time; (b) Although RUB may be a noisy observation of the real state of a node, we expect that incorrect inference of the local observation will be corrected by the mutual influence among all nodes of an invariant network through the iterative process of LBP.

To integrate the prior and local observation into the belief propagation, we need to modify the previously stated instantiation of the node compatibility function Φ. For simplicity, we directly use ω_i as $\Phi(abnormal, \omega_i)$ and $1 - \omega_i$ as $\Phi(normal, \omega_i)$, where ω_i is either initialized to RB or set to RUB. Accordingly, the belief at a node i is proportional to the product of the local observation at that node $\Phi(\lambda, \omega_i)$ and all the messages coming into node i:

$$b_i(\lambda) = k\Phi(\lambda, \omega_i) \prod_{j \in N(i)} m_{ji}(\lambda), \tag{10}$$

And the message updating rule is changed as:

$$m_{ij}(\lambda) = \sum_{\lambda'} \Phi(\lambda, \omega_i)\Psi(\lambda, \lambda') \prod_{n \in N(i) \setminus j} m_{ni}(\lambda'), \tag{11}$$

V. Experimental Results

A. The Experimental Setup

Experimental Data. We test the algorithm on both synthetic and real-world data sets. First, we randomly generate a group of time series, each of which contains 1323 points. We generated invariant networks by using the first 1300 points of all time series. Then we tracked the invariant networks with the remaining 23 observations. We performed metric anomaly ranking at a certain timestamp these 23 observations. To test the scalability of algorithm, we randomly generated 7 groups of time series with different sizes. Specifically, the numbers of time series are 500, 1000, 1500, 2000, 3000, 4000 and 5000. For each group of time series, we will conduct the experimental test and comparison separately.

The real-world monitoring data were collected from a real small bank information system. In total, there are 11 categories, and each category includes different numbers of measurements. This monitoring data is used to calculate various flow intensities with a sampling unit equal to 6 seconds. We have 1273 monitoring (time series or metrics), which will be used in the following experiments. In the experiment, we first collected a set of training data which were collected at the normal state

of the system. Each monitoring data within the training set contains 168 points. We used this training set to search all invariants and generated the invariant network as described in section II. Also, we collected another set of data for these 1273 monitoring data, which were collected during the abnormal state of the system. We have 140 observed points for each monitoring data. We use this set of data as the test set. We track the invariant network with this test set by using the method described in section II. Then, we perform the metric anomaly ranking at different timestamps. In addition, for both real-world and synthetic data, we specify the range for the order $[n, m, k]$ as $0 \leq n, m, k \leq 2$. Also the threshold of fitness score is set as 0.7.

Baseline Method. We compare our algorithm with a baseline method proposed in [1]. It simply used the RB defined in equation 8 to measure the anomaly degree of each node and rank all nodes based the measured degree. To present all comparisons easily, we denote our method and baseline method as lbp and b respectively.

Experimental Platform All algorithms are implemented in Matlab2011a. All the experiments were conducted on a Windows 7(64bit) with Intel Core2 Quad Q6600 and 4.00GB RAM.

B. Benchmark Generation

One fundamental difficulty of this anomaly ranking is that it is extremely difficult to get the benchmark of anomaly from the real-world distributed systems [3], [5]. A lot of system faults may happen in a distributed systems. Though system experts may be able to manually diagnose the system, identify the root causes and resolve the problems, it is almost impossible to ask them to rank all metrics according to anomaly. Furthermore, for a lot of cases, system experts may just reboot the system or machines to solve the problem without any awareness of possible root causes. Thus system experts do want to know the metric anomaly ranking when a fault happens in a distributed system, but they can not provide the ground truth or benchmark for the anomaly ranking.

To this end, in the paper, we propose one method to artificially generate the benchmark with the synthetic data and provide a standard for us to evaluate different anomaly ranking algorithms. Specifically the anomaly of a time series (metric) usually indicates the non-normal change. Based on this intuition, we generate the benchmark of anomaly via measuring the changing ratio of amplitude of time series. If the one observation of one time series is y_1, after randomly changing the observation from y_1 to y_2 we measure manually-injected degree of anomaly as: $d_i = \frac{|y_2 - y_1|}{|y_1|}$. We denote this benchmark generation method as **amplitude-based benchmark**. Given all metrics within the invariant network we select to study, we randomly select a small portion of metrics and manually inject certain degree of anomaly to each one metric and record all anomaly degree of all selected metrics. Then we rank all selected metrics according to this artificial anomaly degree (from high degree to low degree) and treat such as rank as our first benchmark.

For the real-world data used in this paper, we was told about the root cause of system faults. Specifically, the system domain experts have specified that the root cause is related to "DB16", which indicates a particular machine component in the system. Also, this root cause leads to some faults at the observation 120th in the test data. Thus, in the experiment, we would like to leverage this information to evaluate the ranking algorithm. Basically, we will demonstrate how the top abnormal metrics ranked by our algorithm are related to this known root cause.

C. Validation Metrics

Given the benchmark of anomaly ranking, we use Precision, Recall and nDCG [6] to validate the effectiveness of anomaly ranking algorithm. All of these three measurements are widely used in ranking-related works [7], [7], [8]. For nDCG, it is defined as $nDCG_p = \frac{DCG_p}{IDCG_p}$. Here, DCG_p is compute as $DCG_p = \sum_{i=1}^{p} \frac{2^{rel_i} - 1}{\log_2(1+i)}$, where rel_i represents the anomaly degree (benchmark) of the metric at position i of the ranking list. And $IDCG_p$ is the ideal DCG at position p of the ideal ranking list, which can be done by sorting all metrics (all nodes of the examinzed invariant network) by the anomaly degree (benchmark). Typically, we have $p \leq K$ if we examine the quality of top-K metrics in the ranking list.

D. Ranking Performances on Synthetic Data

Before the network tracking, for every synthetic data set and a benchmark generation method, we randomly select 15% of invariant network nodes (metrics) to inject the anomaly. And we measure the anomaly degree for the selected nodes with amplitude-based or residual-based method. Thus for each synthetic data set, we have two types of benchmark ranking of anomaly. Then with each benchmark ranking, we compute the average Precision, Recall and nDCG of the sets of same size data sets.

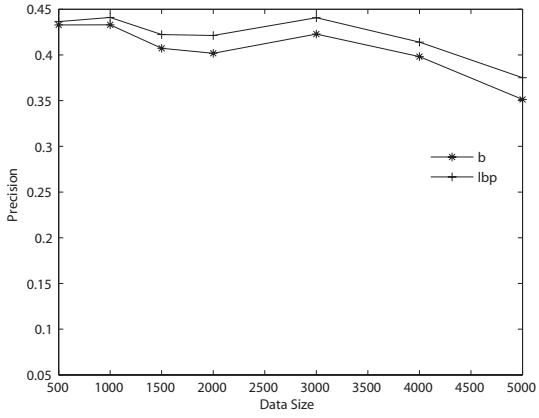

Fig. 3. Precision Comparisons on Different Data Sets with Amplitude-based Benchmark

In Figure 3, we compare algorithms on the Precision of top-K metrics with the amplitude-based benchmark. Note that the computation is based on top-K metrics and we set K as twice as the number of the selected nodes for each benchmark. As we can see in Figure 3, the algorithm developed in this paper can outperform the baseline methods on all data sets with different sizes. In Figure 4, We show the comparison of Recall of top-K metrics based on the amplitude-based benchmark.

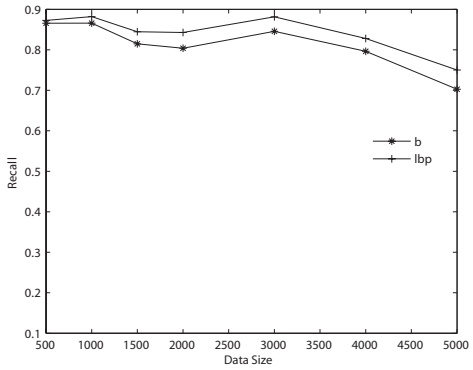

Fig. 4. Recall Comparisons on Different Data Sets with Amplitude-based Benchmark

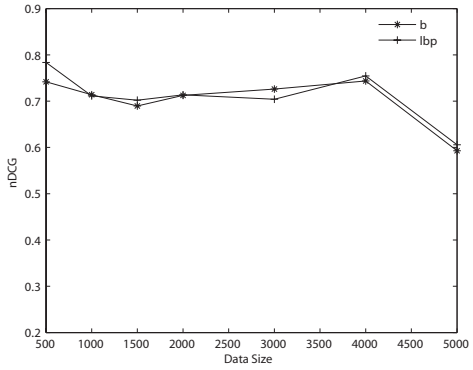

Fig. 5. nDCG Comparisons on Different Data Sets with Amplitude-based Benchmark

In Figure 5, we show the nDCG@p on different data sets with the amplitude-based benchmark. Our algorithm leads to better performance (higher nDCG) than the baseline algorithm.

E. Ranking Performances on Real-world Data

In this subsection, we compare the ranking performances of different algorithms on the real-world data.

In the real-world data, there is a root cause related to the "DB16" at the 120th observation in the test data. It is difficult to ask system operators to provide the rank of abnormal metrics because there is huge dependency among various metrics. But we have a unique semantic label associated with each metric. For example, some semantic labels may be "DB15 DISK hdaw Block" and "WEB26 PAGEOUT RATE". Our goal here is to demonstrate how the top metrics in the ranking list are related to the "DB16"-related root cause. Thus we perform online tracking in invariant networks and do metric ranking at the 120th observation in the test data. Table II shows the label of top-32 metrics generated by different ranking methods. For example, in the ranking list of the baseline algorithm, there are 5 metrics related to "DB16". And the lbplp algorithm produces the most number (i.e., 10) of metrics related to "DB16" in the top-32 metrics. However, it is possible that other metrics, which are not labeled with "AP11", can become abnormal after the root cause happens, because there is much underlying dependent relationship between different components of the information system and such dependency will help to spread the influence of root cause in the information system.

TABLE II. TOP METRICS BY DIFFERENT METHODS

AP11-Related Metrics	**b**	**lbp**
AP11 PACKET Output	DB17 DISK hdm Request	AP13 DISK hd30 Request
AP11 DISK hd0 Request	DB18 DISK hdk Busy	AP11 DISK hd45 Request
AP11 CPU User	DB18 DISK hday Busy	DB16 CPU User
AP11 DISK hd45 Request	DB16 DISK hday Busy	AP11 DISK hd0 Request
AP11 DISK hd45 Busy	DB17 DISK hdm Block	DB18 DISK hdw Request
AP11 DISK hd1 Request	DB16 DISK hdk Busy	AP11 DISK hd45 Block
AP11 CPU System	DB15 DISK hdk Busy	AP11 PACKET Input
AP11 CPU Waitio	DB16 DISK hdm Block	DB16 DISK hday Busy
AP11 DISK hd45 Block	DB15 DISK hday Busy	DB16 DISK hdk Busy
AP11 PACKET Input	DB18 DISK hdbu Request	DB17 DISK hdm Block
AP11 DISK hd0 Busy	AP11 PACKET Input	DB15 DISK hday Busy
AP11 DISK hd30 Busy	AP13 DISK hd30 Request	DB16 DISK hdm Block

VI. CONCLUDING REMARKS

In this paper, we have presented a study of exploiting monitoring data, collected at different points in distributed information systems, for system fault detection and diagnosis. Specifically, we proposed a metric anomaly ranking problem, which is motivated by the complexity arising from the traditional invariant-based modeling of system dynamics. Indeed, based on the domain understanding of system faults, their influence patterns in invariant networks, and link analysis in networks, we developed a metric anomaly ranking algorithm based on belief propagation. This algorithm which integrates local observation well into the message passing process was demonstrated to be able to outperform the baseline method with both real-world and synthetic data.

VII. ACKNOWLEDGEMENTS

This research was supported in part by National Natural Science Foundation of China under Grant 61203034 and Grant 61373046.

REFERENCES

[1] G. Jiang, H. Chen, and K. Yoshihira, "Discovering likely invariants of distributed transaction systems for autonomic system management," in *proceedings of the 3rd International Conference on Autonomic Computing*, 2006, pp. 199–208.

[2] G. Jiang, H. Chen, and K. Yeshihira, "Efficient and scalable algorithms for inferring likely invariants in distributed systems," *Transactions on Knowledge and Data Engineering*, vol. 19, no. 11, pp. 1508–1523, 2007.

[3] G. Jiang, H. Chen, and K. Yoshehira, "Modeling and tracking of transaction flow dynamics for fault detection in complex systems," *IEEE Transactions on Dependable and Secure Computing*, vol. 3(4), pp. 312–326, 2006.

[4] L. Ljung, *System Identification: Theory for the User*, 2nd ed. Prentice Hall PTR, 1998.

[5] S. Ghanbari and C. Amza, "Semantic-driven model composition for accurate anomaly diagnosis," in *proceedings of the 24th International Conference on Software Engineering*, 2002, pp. 291–301.

[6] K. Jarvelin and J. Kekalainen, "Cumulated gain-based evaluation of ir techniques," *ACM Transactions on Information Systems*, vol. 20, no. 4, pp. 422–446, 2002.

[7] H. Valizadegan, R. Jin, R. Zhang, and J. Mao, "Learning to rank by optimizing ndcg measure," in *proceedings of the 23rd Annual Conference on Neural Information Processing Systems*, 2009.

[8] S. Wei, Y. Zhao, Z. Zhu, and N. Liu, "Multimodal fusion for video search reranking," *Transactions on Knowledge and Data Engineering*, vol. 22, no. 8, pp. 1191–1199, 2010.