Ranking Causal Anomalies by Modeling Local Propagations on Networked Systems

Conference Paper · November 2017

DOI: 10.1109/ICDM.2017.129

7 authors, including:

Wei Cheng
University of North Carolina at Chapel Hill
62 PUBLICATIONS **788** CITATIONS
SEE PROFILE

Kai Zhang
Suzhou Institute of Biomedical Engineering and Technology, Chinese Academy of Sc…
227 PUBLICATIONS **6,408** CITATIONS
SEE PROFILE

Ranking Causal Anomalies by Modeling Local Propagations on Networked Systems

***Abstract*—Complex systems are prevalent in many fields such as finance, security and industry. A fundamental problem in system management is to perform diagnosis in case of system failure such that the causal anomalies, i.e., root causes, can be identified for system debugging and repair. Recently, invariant network has proven a powerful tool in characterizing complex system behaviors. In an invariant network, a node represents a system component, and an edge indicates a stable, significant interaction between two components. During the evolution of the invariant network, an edge may break at some time point when either of its end nodes is affected by system faults. Therefore, the content of the invariant network, in particular the evolutionary patterns of the broken edges, can serve as an important clue in locating causal anomalous nodes. Recent approaches have shown that by modeling fault propagation underlying the broken edges in the invariant network, causal anomalies can be effectively discovered. Despite their success, the existing methods have a major limitation: they typically assume there is only a single and global fault propagation in the entire network. However, in real-world large-scale complex systems, it's more common for *multiple* fault propagations to grow simultaneously and *locally* within different node clusters and jointly define the system failure status. Inspired by this key observation, we propose a two-phase framework to identify and rank causal anomalies in a fine-grained manner. In the first phase, a probabilistic clustering is performed to uncover impaired node clusters in the invariant network. Then, in the second phase, a principled low-rank network diffusion model is designed to backtrack causal anomalies in different impaired clusters. Experiments on real-world bank information system and power plant sensor datasets show that our method consistently outperforms the state-of-the-art methods by a large margin.**

I. Introduction

Complex systems are ubiquitous in modern manufacturing industry and information services. Monitoring behaviors of these large-scale systems generates massive log data, such as the metric readings from the networked sensors distributed in a power plant, and the flow intensities of system logs from the cloud computing facilities in Google, Yahoo! and Amazon [1]. The unprecedented growth of the monitoring data increases the demand for automatic system managements [2]. A central task in managing these complex systems is to detect anomalies and diagnose system faults. It has been reported that 1 minute of downtime in an automotive manufacturing plant could result in as much as $20,000 cost [3]. Hence a timely diagnosis of system faults is crucial to avoid serious money waste and business loss.

Due to its practical importance, there have been intensive interests in developing algorithms to infer whether there is a system failure at a time and if yes, which system components (i.e., basic system units) are causal anomalies (or root causes). An early approach is to examine individual time series recorded on system components and infer anomalies by a thresholding method [4]. However, in practice it is difficult to set a proper threshold due to the dynamics and heterogeneity of the data. More effective and recent approaches typically start with building system profiles using historical time series data, and then detect anomalies via analyzing patterns in the profiles [5], [6].

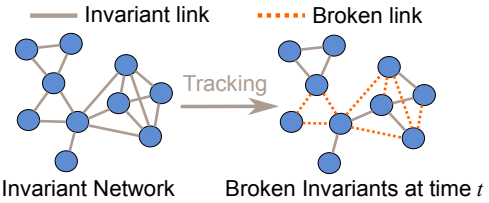


Fig. 1. Invariant network and broken invariants.

The *invariant network* model is a successful example in profiling complex system behaviors [5], [7], [1], [2], [8], [9], [10]. Its focus is to discover stable and significant dependencies between pairs of system components that are monitored through time series recordings, so as to profile system status for downstream reasonings. A strong dependency between a pair of components is called an *invariant* relationship. By combining the invariants learned from all monitoring components, an *invariant network* can be constructed. As illustrated in Fig. 1 (left), in an invariant network, a node represents a system component. A solid line represents an invariant link/relationship between a pair of components.

The practical value of an invariant network is that it can shed important light on abnormal system behaviors and in particular the source of anomalies, by checking whether existing invariants are broken. In Fig. 1 (right), the dotted lines indicate the invariant links are broken at time point t. Such a broken invariant link usually implies abnormal behaviors have occurred in one or both of its connected components [5]. A network including all system components and all the broken invariant links at a given time is called a *broken network*. For example, in Fig. 1 (right), a broken network will contain all nodes and only those dotted lines.

With the use of invariant and broken networks, several ranking algorithms were developed to diagnose system faults. For example, the method in [2] determines causal anomalies by the percentage of broken invariants within the neighborhood of each node. However, this percentage can be easily biased by fake broken invariants, which occur frequently due

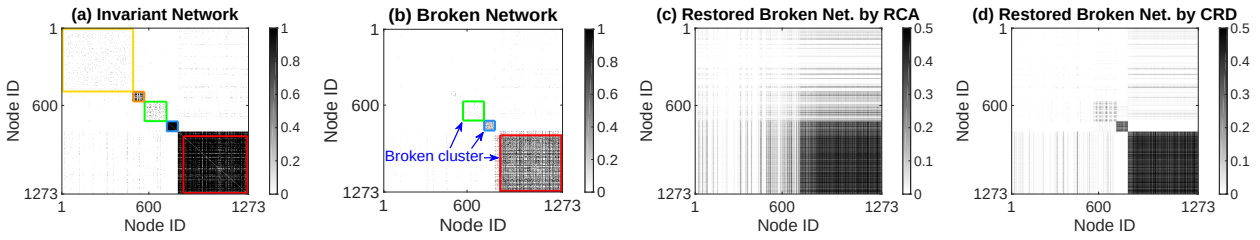


Fig. 2. The adjacency matrices of invariant and broken networks from a bank information system dataset.

to environmental noises in complex systems. More recently, researchers found system faults are seldom isolated. Instead, starting from the root nodes, anomalous behavior will propagate to neighboring nodes in a cascading manner [5]. Such observations have led to a number of successful examples in modeling fault propagations [8], [10]. Despite their success in identifying some causal anomalies, a major limitation of these approaches lies in their basic assumption. They typically assume there is a single and global propagation over the entire network, which is not precise in many emerging applications.

Fig. 2(a) and 2(b) show the adjacency matrices of an invariant network and its corresponding broken network at a system failure time point from a real-world bank information system. Each entry in Fig. 2(a) represents an invariant link. Each entry in Fig. 2(b) represents a broken link. In Fig. 2(a), we can observe that there are several densely connected clusters, e.g., functional modules, in the invariant network. Among them, three clusters highlighted by red, blue and green can also be observed in the broken network in Fig. 2(b). In other words, these three clusters are heavily impaired. It should be noted that different clusters have few connections in between. This means it will be difficult for system faults to propagate across different clusters.

The above application illustrates some important properties of system fault propagations, which have not been taken into account by the existing methods: (1) system faults are propagated *locally* within different clusters, rather than traversing globally through the whole network; (2) there can be *multiple* fault propagations spreading in parallel in different clusters in the system. Therefore, by assuming a single and global propagation in the network, the existing methods cannot locate multiple impaired clusters. Consequently, many true anomalous nodes cannot be accurately detected. For example, one recent algorithm RCA [10] uses a network reconstruction based method to model fault propagation. Fig. 2(c) shows the reconstructed broken network of RCA on the bank information system dataset. From the figure, we can see RCA confuses the blue cluster with the red one and misses the green cluster.

To address the limitations of the existing methods, in this paper, we propose the **C**luster **R**anking based fault **D**iagnosis (CRD) algorithm to rank causal anomalies in a fine-grained manner. CRD is a two-phase framework. In Phase I, it identifies and ranks clusters in the invariant network by their severities of impairments. To enhance the accuracy of cluster finding, a joint clustering scheme is designed to leverage the complementary information in invariant and broken networks. In Phase II, a diffusion based low-rank network reconstruction model is proposed to backtrack causal anomalies in impaired clusters found in Phase I. This model can capture local and paralleled fault propagations in different clusters, making it suitable for locating multiple causal anomalies. Moreover, we provide rigid theoretical analysis on CRD algorithm, showing its convergence and complexity. Fig. 2(d) shows the reconstructed broken network by CRD. It can be seen that CRD can successfully uncover all the broken clusters.

In practice, causal anomalies often remain stable within a short period of time when the faults are propagating in the system [10]. Thus we can jointly model successive broken networks to resolve ambiguities caused by system noise. While many existing methods usually consider one broken network at a time, CRD can integrate multiple successive broken networks for more reliable fault diagnosis.

In our experiments, we compare CRD with the state-of-the-art methods on real-world datasets including a bank information system dataset and a power plant sensors dataset. The comprehensive results on real-life datasets demonstrate that CRD consistently outperforms the alternatives.

II. Preliminaries and Problem Definition

In this section, we introduce the technique of the invariant network model [5], and then describe the problem setting.

A. Invariant Network and Broken Invariants

The *invariant* model is used to uncover significant pairwise relationships among massive set of time series. Let $x(t)$ and $y(t)$ be a pair of time series under consideration, such as two sensor readings on two system components, where t is the time index, then their relationship can be described by a linear regression function according to the AutoRegressive eXogenous (ARX) model [11]:

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

where $[n, m]$ is the *order* of the model, which determines how many previous steps are affecting the current output. k is a time delay factor between x and y. Parameters a_i and b_j indicate how strongly a previous step is impacting the current output, which can be learned by the least-square fitting of Eq. (1) to the training data. In real-world applications such as anomaly detection in physical systems, $0 \leq m, n, k \leq 2$ is a popular choice [5], [1].

Let $\boldsymbol{\theta} = \{a_1, ..., a_n, b_0, ..., b_m\}$ be the model parameters, after it is obtained, the prediction of $y(t)$ can be found using Eq. (1) by feeding $\boldsymbol{\theta}$ and observations $y(t-1), ..., y(t-n), x(t-k), ..., x(t-k-m)$. Let $\hat{y}(t, \boldsymbol{\theta})$ represent the prediction, once it is obtained, a *fitness score* $F(\boldsymbol{\theta})$ [12] is used to evaluate how well the learned model $\boldsymbol{\theta}$ fits the real observations as

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

where N and $\bar{y}$ are the length and mean of the time series $y(t)$, respectively. A large fitness score indicates a better fitting of the model. Then, an invariant is declared on a pair of times series x and y if the fitness score is larger than a pre-defined threshold. A network including all the invariant links is called an *invariant network*.

After training the invariant model, each invariant will be tracked using a normalized residual $R(t)$ [12], [1]:

$$R(t) = |y(t) - \hat{y}(t, \boldsymbol{\theta})| / \varepsilon_{\max} \tag{3}$$

where $\varepsilon_{\max} = \max_{1 \le t \le N} |y(t) - \hat{y}(t, \boldsymbol{\theta})|$ is the maximal error. If the residual exceeds a prefixed threshold, then the invariant is declared as "broken", i.e., the corresponding dependency relationship vanishes. At time $t = T_b$, A network including all nodes in the invariant network and all broken edges is called a *broken network* at time T_b.

B. Problem Description

We represent the invariant network and broken network by their corresponding adjacency matrices $\mathbf{A} \in \{0, 1\}^{n \times n}$ and $\mathbf{B} \in \{0, 1\}^{n \times n}$, where n is the number of nodes (e.g., system components) in the system. The two matrices can be obtained as discussed in Sec. II-A. An entry $\mathbf{A}_{xy}$ equals 1 indicates an invariant dependency exists between nodes x and y; 0 otherwise; and an entry $\mathbf{B}_{xy}$ equals 1 indicates the invariant link between nodes x and y is broken; 0 otherwise. It is worth to mention that the proposed CRD algorithm also allows $\mathbf{A}$ and $\mathbf{B}$ to contain continuous entries. In this case, $\mathbf{A}_{xy}$ and $\mathbf{B}_{xy}$ can be weighted by fitness score $F(\boldsymbol{\theta})$ (Eq. (2)) and residual $R(t)$ (Eq. (3)), respectively.

The goal of this work is to detect abnormal nodes in invariant network $\mathbf{A}$ that are most likely to be the causes of the broken edges in $\mathbf{B}$. Since such anomalies may exist in multiple clusters, we call them *multifaceted causal anomalies*. Accurately detecting multifaceted causal anomalies will be extremely useful for debugging complex system problems that are jointly defined by different impaired functional modules (i.e., broken node clusters).

III. The CRD Algorithm

In this section, we introduce our CRD algorithm, which is a two-phase framework. In Phase I, CRD ranks and identifies node clusters by their severities of impairments. In Phase II, it backtracks causal anomalies by modeling multiple local fault propagations in different broken clusters. It is worth to mention that, existing methods are unaware of the clustering structures of the invariant network and broken network.

A. Phase I: Broken Cluster Identification

First, we propose a probabilistic clustering model to jointly cluster invariant network and broken network, and in the meantime, rank broken clusters. The intuition for the joint clustering is that, a set of nodes that work coordinately in normal status and break concurrently in abnormal status are more likely to be in the same cluster. Therefore, jointly clustering the two networks will be useful to enhance the accuracy of finding broken clusters.

The Basic Clustering Method. We adopt the doubly stochastic matrix decomposition as the basic method to cluster an invariant network due to its superior performance on sparse networks [13], which is introduced as following.

Suppose there are k clusters in an invariant network $\mathbf{A}$, let $\mathbf{U} \in \mathbb{R}_{+}^{n \times k}$ be a cluster membership matrix with $\mathbf{U}_{xi} = P(i|x)$ indicating the probability that node x belongs to cluster i. Then a doubly stochastic approximation to $\mathbf{A}$ is defined by

$$\hat{\mathbf{A}}_{xy} = \sum_{i=1}^{k} \frac{\mathbf{U}_{xi} \mathbf{U}_{yi}}{\sum_{z=1}^{n} \mathbf{U}_{zi}} \tag{4}$$

where i is the cluster index, x, y and z are node indexes. Note $\hat{\mathbf{A}} \in \mathbb{R}_{+}^{n \times n}$ is symmetric and both of its columns and rows sum up to 1. Therefore, it is referred to as *doubly stochastic*.

The clustering problem is to infer $\mathbf{U}$ by minimizing the approximation error of the KL-Divergence $\mathcal{D}_{KL}(\mathbf{A} || \hat{\mathbf{A}})$. After removing some constants, this is equivalent to minimize

$$- \sum_{(x,y) \in \mathcal{E}_A} \mathbf{A}_{xy} \log \hat{\mathbf{A}}_{xy} \tag{5}$$

where $\mathcal{E}_A$ represents the set of all edges in network $\mathbf{A}$.

To provide control of the sparsity of $\mathbf{U}$, a Dirichlet prior on $\mathbf{U}$ can be introduced [13], which gives the following objective function for individual network clustering

$$\begin{aligned} \mathcal{J}_A(\mathbf{U}) = & - \sum_{(x,y) \in \mathcal{E}_A} \mathbf{A}_{xy} \log \hat{\mathbf{A}}_{xy} - (\alpha - 1) \sum_{xi} \log \mathbf{U}_{xi} \\ & \text{s.t. } \mathbf{U} \ge 0, \ \mathbf{U}\mathbf{1}_k = \mathbf{1}_n \end{aligned} \tag{6}$$

where α ($\alpha \ge 1$) is a parameter in the Dirichlet distribution, large α usually results in more non-zero entries in $\mathbf{U}$. $\mathbf{1}_k$ is a column vector of length k with all 1's. The equality constraints are enforced to preserve the probabilistic interpretation of $\mathbf{U}_{xi}$.

Ranking Broken Clusters. Next, we develop a method to rank clusters by their broken severities. Our method uses a generative process to model broken invariants in $\mathbf{B}$. The intuition is that, if two nodes x and y reside in the same severely broken cluster, the invariant link (x, y) is more likely to break. Here, we need a metric to quantify how severe a cluster is broken. Thus for each cluster i in the invariant network, we define an unknown *broken score* as $\mathbf{s}_i$ ($0 \le \mathbf{s}_i \le 1$). A higher $\mathbf{s}_i$ means a more severely broken cluster i.

To evaluate how likely an invariant link (x, y) will break, we need a probability for this event. According to the above intuition, this probability should satisfy two criteria: (1) within $[0, 1]$; and (2) it is large only if nodes x and y belong to the

same cluster(s) i and cluster(s) i has a high broken score $\mathbf{s}_i$. Therefore, we propose to use

$$P_b(x,y) = \sum_{i=1}^{k} \mathbf{U}_{xi}\mathbf{U}_{yi}\mathbf{s}_i \tag{7}$$

as the broken probability of an invariant (x,y). It is easy to verify $P_b(x,y)$ satisfies the above two criteria. Then, to model the sparse occurrences of broken edges, we follow the convention of modeling sparse networks [14] and use Bernoulli distribution to model the generation of a broken invariant (x,y) by

$$\mathbf{B}_{xy} \sim \text{Bernoulli}\big(P_b(x,y)\big) \tag{8}$$

Let $\mathcal{E}_B$ be the set of all edges in $\mathbf{B}$, then the probability to collectively generate a broken network is

$$P(\mathbf{B}|\mathbf{U},\mathbf{s}) = \prod_{(x,y)\in\mathcal{E}_B} P_b(x,y) \prod_{(x,y)\in\mathcal{E}_A\setminus\mathcal{E}_B} [1-P_b(x,y)] \tag{9}$$

Let $\mathbf{W} \in \{0,1\}^{n\times n}$ be an indicator matrix, with $\mathbf{W}_{xy}=1$ iff $(x,y)\in\mathcal{E}_A\setminus\mathcal{E}_B$, i.e., (x,y) is a non-broken invariant link. Then we can write the negative log-likelihood function as

$$\begin{aligned}\mathcal{J}_B(\mathbf{U},\mathbf{s}) = &-\sum_{xy}\mathbf{B}_{xy}\log\big(\sum_i \mathbf{U}_{xi}\mathbf{U}_{yi}\mathbf{s}_i\big)\\ &-\sum_{xy}\mathbf{W}_{xy}\log\big(1-\sum_i \mathbf{U}_{xi}\mathbf{U}_{yi}\mathbf{s}_i\big)\end{aligned} \tag{10}$$

which is our objective for learning to rank broken clusters. Here, the variable $\mathbf{s}_i$ serves as the ranking score. In Sec. III-C, we will introduce its use for enhancing anomaly detection.

A Unified Objective Function. As discussed in the beginning of this section, to leverage the complementary information in invariant and broken networks, we integrate $\mathcal{J}_A$ in Eq. (6) and $\mathcal{J}_B$ in Eq. (10) into a joint optimization problem

$$\begin{aligned}&\min_{\mathbf{U},\mathbf{s}}\ \mathcal{J}_{CR}(\mathbf{U},\mathbf{s}) = \mathcal{J}_A + \beta\mathcal{J}_B\\ &\text{s.t. } \mathbf{U}\mathbf{1}_k = \mathbf{1}_n,\ \mathbf{U}\geq 0,\ 0\leq \mathbf{s}_i\leq 1,\ \forall 1\leq i\leq k\end{aligned} \tag{11}$$

where β is a parameter to balance $\mathcal{J}_A$ and $\mathcal{J}_B$. Intuitively, the more reliable the broken network, the larger the β.

B. Phase II: Causal Anomaly Ranking

To infer causal anomalous nodes, we consider the very practical scenario of fault propagation, namely anomalous system status can always be traced back to a set of initial seed nodes, i.e., causal anomalies. These anomalies can propagate along the invariant network, most probably towards neighbors via paths represented by the invariant links in $\mathbf{A}$. To model this process, we employ the label propagation technique [15]. Suppose there is an unknown *seed vector* $\mathbf{e}\in\mathbb{R}_+^{n\times 1}$ with $\mathbf{e}_x$ denoting the degree that node x is a causal anomaly. After propagation, each node x will obtain a *status score* $\mathbf{r}_x$ to indicate to what extent it is impacted by the causal anomalies. Then the propagation from $\mathbf{e}$ to $\mathbf{r}$ can be modeled by the following optimization problem

$$\begin{aligned}\min_{\mathbf{r}\geq 0}\ & c\sum_{x,y=1}^{n}\mathbf{A}_{xy}\Big(\frac{\mathbf{r}_x}{\sqrt{\mathbf{D}_{xx}}}-\frac{\mathbf{r}_y}{\sqrt{\mathbf{D}_{yy}}}\Big)^2 + (1-c)\sum_{x=1}^{n}(\mathbf{r}_x-\mathbf{e}_x)^2\\ &= c\mathbf{r}^T(\mathbf{I}_n-\tilde{\mathbf{A}})\mathbf{r} + (1-c)\|\mathbf{r}-\mathbf{e}\|_F^2\end{aligned} \tag{12}$$

where $\mathbf{I}_n$ is an $n\times n$ identity matrix, $\tilde{\mathbf{A}} = \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$ is a symmetrically normalized matrix of $\mathbf{A}$, and $\mathbf{D}$ is a diagonal matrix with $\mathbf{D}_{xx}=\sum_{y=1}^{n}\mathbf{A}_{xy}$.

The first term in Eq. (12) encourages neighboring nodes to have similar status scores, and the second term penalizes large bias from the initial seeds. c $(0<c<1)$ is a parameter balancing the two terms. It can be verified that the closed-form solution to Eq. (12) is

$$\mathbf{r} = (1-c)(\mathbf{I}_n - c\tilde{\mathbf{A}})^{-1}\mathbf{e} \tag{13}$$

which establishes an explicit relationship between $\mathbf{r}$ and $\mathbf{e}$.

As discussed in Sec. I, in real-world applications, causal anomalies often propagate their impacts inside their associated clusters. Thus for each cluster i, we define $\mathbf{e}^{(i)}\in\mathbb{R}_+^{n\times 1}$ as a *cluster-specific* seed vector. Moreover, instead of directly using $\mathbf{e}_x^{(i)}$ as the causal anomaly score of node x, we use $\mathbf{U}_{xi}\mathbf{e}_x^{(i)}$, where $\mathbf{U}_{xi}$ is obtained in Phase I, to emphasize that, node x is a causal anomaly of cluster i if it resides in cluster i (with a large $\mathbf{U}_{xi}$ value) and is abnormal (with a large $\mathbf{e}_x^{(i)}$ value).

Correspondingly, different clusters will have different status score vectors $\mathbf{r}^{(i)}\in\mathbb{R}_+^{n\times 1}$. Then the propagation relationship between $\mathbf{e}^{(i)}$ and $\mathbf{r}^{(i)}$ can be represented by

$$\mathbf{r}^{(i)} = (1-c)(\mathbf{I}_n - c\tilde{\mathbf{A}})^{-1}(\mathbf{U}_{*i}\circ\mathbf{e}^{(i)}) \tag{14}$$

where $\circ$ is entry-wise product, $\mathbf{U}_{*i}$ is the i^{th} column of $\mathbf{U}$.

To exploit the broken edge pattern, we propose to use $\{\mathbf{r}^{(i)}\}_{i=1}^{k}$ to reconstruct the broken network $\mathbf{B}$. The intuition is as following. When an invariant link (x,y) is broken, i.e., $\mathbf{B}_{xy}$ is large, then at least one node of x and y should be perturbed by some causal anomalies from some clusters. That is, either $\mathbf{r}_x^{(i)}$ or $\mathbf{r}_y^{(i)}$ is large for some cluster i. This suggests a reconstruction error as

$$\sum_{(x,y)\in\mathcal{E}_A}\Big(\sum_{i=1}^{k}\mathbf{r}_x^{(i)}\mathbf{r}_y^{(i)} - \mathbf{B}_{xy}\Big)^2 \tag{15}$$

Let $\mathbf{E}=[\mathbf{e}^{(1)},...,\mathbf{e}^{(k)}]$, $\mathbf{R}=[\mathbf{r}^{(1)},...,\mathbf{r}^{(k)}]$, and $\mathbf{H}=(1-c)(\mathbf{I}_n-c\tilde{\mathbf{A}})^{-1}$, from Eq. (14), we have $\mathbf{R}=\mathbf{H}(\mathbf{U}\circ\mathbf{E})$. Then, let $\mathbf{C}\in\{0,1\}^{n\times n}$ be an indicator matrix with $\mathbf{C}_{xy}=1$ iff $(x,y)\in\mathcal{E}_A$, we can rewrite Eq. (15) by a matrix form and obtain the following objective function

$$\min_{\mathbf{E}\geq 0}\ \mathcal{J}_H = \|\mathbf{C}\circ[\mathbf{H}(\mathbf{U}\circ\mathbf{E})(\mathbf{U}\circ\mathbf{E})^{\mathrm{T}}\mathbf{H}^{\mathrm{T}}] - \mathbf{B}\|_F^2 + \tau\|\mathbf{E}\|_1 \tag{16}$$

Here, an ℓ_1 norm on $\mathbf{E}$ is added to encourage sparsity of $\mathbf{E}$ because practically often a few nodes could be causal anomalies. τ is a controlling parameter, a larger τ typically results in more zeros in $\mathbf{E}$.

C. Ranking with Unified Scores

To integrate the results from Phase I and II, we propose a unified causal anomaly score $\mathbf{f}_x$ for each node x. Ideally, this score should place more priority to a node x if it is a causal anomaly to some cluster i (with large $\mathbf{U}_{xi}\mathbf{e}_x^{(i)}$) and cluster i is broken severely (with large $\mathbf{s}_i$). This suggests a simple form $\mathbf{f}_x = \mathbf{U}_{xi}\mathbf{e}_x^{(i)}\mathbf{s}_i$. Equivalently, the score vector $\mathbf{f}$ is

$$\mathbf{f} = (\mathbf{U}\circ\mathbf{E})\mathbf{s} \tag{17}$$

To summarize, in our CRD algorithm, we first optimize $\mathcal{J}_{CR}$ in Eq. (11) to solve $\mathbf{U}$ and $\mathbf{s}$ in Phase I, then plug the obtained $\mathbf{U}$ into $\mathcal{J}_H$ in Eq. (16) and solve $\mathbf{E}$. Finally, all nodes are sorted using $\mathbf{f}$ in Eq. (17), with most suspicious nodes on the top. Alg. 1 summarizes the proposed CRD algorithm, which will be explained in detail in next section.

D. The Learning Algorithm

Learning Algorithm for Phase I. The objective function in Eq. (11) is not jointly convex in $\mathbf{U}$ and $\mathbf{s}$, hence we take an alternating minimization framework that alternately solves $\mathbf{U}$ and $\mathbf{s}$ until a stationary point is achieved.

First, to solve $\mathbf{U}$, we use an Auxiliary Function approach [16] to derive a multiplicative updating rule. Before presenting the solution, we need to introduce several notations. Let $\mathcal{J}_{CR}(\mathbf{U})$ be the objective function in Eq. (11) w.r.t. $\mathbf{U}$, then the Lagrangian function of $\mathcal{J}_{CR}$ w.r.t. $\mathbf{U}$ is

$$\mathcal{L}_U(\mathbf{U}, \boldsymbol{\lambda}) = \mathcal{J}_{CR}(\mathbf{U}) + \sum_{x=1}^{n} \lambda_x (\sum_{i=1}^{k} \mathbf{U}_{xi} - 1) \tag{18}$$

where $\boldsymbol{\lambda} = (\lambda_1, ..., \lambda_n)^T$ are the Lagrangian multipliers.

Let the gradient of $\mathcal{J}_{CR}(\mathbf{U})$ w.r.t. $\mathbf{U}$ be $\nabla_U = (\nabla_U)^+ - (\nabla_U)^-$, where $(\nabla_U)^+$ and $(\nabla_U)^-$ represent the positive and non-positive parts of ∇_U, respectively. Then the following theorem summarizes the solution. For clarity, detailed theoretical analysis will be presented in the next section.

Theorem 1. *Let* $\lambda_x = (\mathbf{b}_x - 1)/\mathbf{a}_x$ *where*

$$\mathbf{a}_x = \sum_{i=1}^{k} \frac{\mathbf{U}_{xi}}{(\nabla_U)^+_{xi}}, \quad \mathbf{b}_x = \sum_{i=1}^{k} \mathbf{U}_{xi} \frac{(\nabla_U)^-_{xi}}{(\nabla_U)^+_{xi}} \tag{19}$$

It holds that $\mathcal{L}_U(\mathbf{U}^{(new)}, \boldsymbol{\lambda}) \leq \mathcal{L}_U(\mathbf{U}, \boldsymbol{\lambda})$*, by updating* $\mathbf{U}$ *according to Eq.* (20)*.*

$$(\mathbf{U}_{xi})^{(new)} \leftarrow \mathbf{U}_{xi} \frac{\mathbf{a}_x (\nabla_U)^-_{xi} + 1}{\mathbf{a}_x (\nabla_U)^+_{xi} + \mathbf{b}_x} \tag{20}$$

From Theorem 1, we can iteratively apply Eq. (20) to decrease the objective value of Eq. (11). In this process, $\mathbf{U}$ will be automatically adjusted to satisfy the equality constraint in Eq. (11). Note Eq. (20) can be directly applied without explicitly specifying the value of $\boldsymbol{\lambda}$ in the algorithm, since the definition of $\boldsymbol{\lambda}$ has already been embedded into Eq. (20).

To solve $\mathbf{s}$, we use a similar approach as before involving Karush-Kuhn-Tucker (KKT) conditions [17]. We denote $\mathcal{J}_{CR}(\mathbf{s})$ as the objective function in Eq. (11) w.r.t. $\mathbf{s}$. Similarly, let the gradient of $\mathcal{J}_{CR}(\mathbf{s})$ w.r.t. $\mathbf{s}$ as $\nabla_s = (\nabla_s)^+ - (\nabla_s)^-$. Then the following theorem presents the solution to $\mathbf{s}$.

Theorem 2. *Fixing other variables, updating* $\mathbf{s}$ *according to Eq.* (21) *monotonically decreases the objective value in Eq.* (11) *until convergence.*

$$\mathbf{s}_i \leftarrow \min\left(\mathbf{s}_i [(\nabla_s)^-_i / (\nabla_s)^+_i], 1\right) \tag{21}$$

Therefore, by alternating between Eq. (20) and Eq. (21), the optimization problem in Phase I is solved.

Learning Algorithm for Phase II. In Phase II, before solving $\mathbf{E}$ for Eq. (16), matrix $\mathbf{H}$ should be pre-computed, which involves a time consuming $O(n^3)$ matrix inversion (see Eq. (16)). To avoid that, we can employ the following lemma.

Algorithm 1: CRD

Input: (1) Adjacency matrices of invariant network $\mathbf{A}$ and broken network $\mathbf{B}$; (2) Logic matrices $\mathbf{W}$ (see Eq. (10)) and $\mathbf{C}$ (see Eq. (16)); (3) Number of clusters k; (4) Parameters $\{\alpha, \beta, c, \tau\}$

Output: $\mathbf{U}$, $\mathbf{s}$, $\mathbf{E}$ and $\mathbf{f}$

Randomly initialize $\mathbf{U}$ and normalize it s.t. $\mathbf{U}\mathbf{1}_k = \mathbf{1}_n$;
Randomly initialize $\mathbf{s}$ and normalize it s.t. $\sum_{x=1}^{n} \mathbf{s}_x = 1$;
repeat
  Update $\mathbf{U}$ by Eq. (20);
  Update $\mathbf{s}$ by Eq. (21);
until *Convergence*
Symmetrical normalization: $\tilde{\mathbf{A}} = \mathbf{D}^{-\frac{1}{2}} \mathbf{A} \mathbf{D}^{-\frac{1}{2}}$;
Initialize $\hat{\mathbf{H}} = \mathbf{I}_n$;
Update $\hat{\mathbf{H}} \leftarrow c\tilde{\mathbf{A}}\hat{\mathbf{H}} + (1-c)\mathbf{I}_n$ until convergence;
Randomly initialize $\mathbf{E}$ with values within $(0, 1]$;
Update $\mathbf{E}$ by Eq. (23) until convergence;
Unified causal anomaly score vector $\mathbf{f} = (\mathbf{U} \circ \mathbf{E})\mathbf{s}$;
return $\mathbf{U}$, $\mathbf{s}$, $\mathbf{E}$ and $\mathbf{f}$.

Lemma 1. *[15] Set* $\hat{\mathbf{H}}^{(0)} = \mathbf{I}_n$*. As* $t \to \infty$*,* $\hat{\mathbf{H}}^{(t)}$ *converges to* $\mathbf{H}$ *by iteratively updating* $\hat{\mathbf{H}}^{(t)}$ *by Eq.* (22)

$$\hat{\mathbf{H}}^{(t+1)} \leftarrow c\tilde{\mathbf{A}}\hat{\mathbf{H}}^{(t)} + (1-c)\mathbf{I}_n \tag{22}$$

The complexity of Eq. (22) is $O(mn)$, where m and n are the number of edges and nodes in $\tilde{\mathbf{A}}$. When $\tilde{\mathbf{A}}$ is sparse, it reduces to $O(n^2)$. Thus $\hat{\mathbf{H}}$ can be computed efficiently.

Now, with the pre-computed $\hat{\mathbf{H}}$, we use the Auxiliary Function approach [16] to derive the solution to $\mathbf{E}$, which is summarized by the following theorem.

Theorem 3. *Updating* $\mathbf{E}$ *by Eq.* (23) *monotonically decreases the objective value in Eq.* (16) *until convergence.*

$$\mathbf{E}_{xi} \leftarrow \mathbf{E}_{xi} \left(\frac{4(\boldsymbol{\Phi} \circ \mathbf{U})_{xi}}{4(\boldsymbol{\Theta} \circ \mathbf{U})_{xi} + \tau} \right)^{\frac{1}{4}} \tag{23}$$

where $\boldsymbol{\Theta} = \hat{\mathbf{H}}^T \{\mathbf{C} \circ [\hat{\mathbf{H}}(\mathbf{U} \circ \mathbf{E})(\mathbf{U} \circ \mathbf{E})^T \hat{\mathbf{H}}^T]\} \hat{\mathbf{H}}(\mathbf{U} \circ \mathbf{E})$ *and* $\boldsymbol{\Phi} = \hat{\mathbf{H}}^T (\mathbf{B} \circ \mathbf{C}) \hat{\mathbf{H}} (\mathbf{U} \circ \mathbf{E})$.

Based on Theorem 1, 2 and 3, we develop the iterative multiplicative algorithm CRD, as summarized in Alg. 1.

E. Theoretical Analysis

Convergence Analysis. Next, we use the Auxiliary Function approach [16] to prove the convergence of Eq. (23) in Theorem 3. Proofs of Theorem 1 and 2 are based on a similar idea. For clarity, we defer them to an online Supplementary Material[1], which also includes the definitions of gradients ∇_U and ∇_s.

Definition 1. *[16] A function* $Z(h, \tilde{h})$ *is an* **auxiliary function** *for a given function* $J(h)$ *if the conditions* $Z(h, \tilde{h}) \geq J(h)$ *and* $Z(h, h) = J(h)$ *are satisfied.*

The following theorem presents the auxiliary function for $\mathcal{J}_H$ in Eq. (16).

[1] https://www.dropbox.com/s/9l7oexi9ug55ux8/crdsup.pdf?dl=0

Theorem 4. *Let $\mathcal{J}_H(\mathbf{E})$ denote the sum of all terms in Eq.* (16) *that contains* $\mathbf{E}$*, then the following function*

$$\begin{aligned}\mathcal{Z}_E(\mathbf{E},\tilde{\mathbf{E}}) = &-2\sum_{xi}(\tilde{\mathbf{\Phi}}\circ\mathbf{U})_{xi}\tilde{\mathbf{E}}_{xi}(1+\log\frac{\mathbf{E}_{xi}\mathbf{E}_{yi}}{\tilde{\mathbf{E}}_{xi}\tilde{\mathbf{E}}_{yi}})\\ &+\sum_{xi}(\tilde{\mathbf{\Theta}}\circ\mathbf{U})_{xi}\frac{\mathbf{E}_{xi}^4}{\tilde{\mathbf{E}}_{xi}^3}+\frac{\tau}{4}\sum_{xi}\frac{\mathbf{E}_{xi}^4+3\tilde{\mathbf{E}}_{xi}^4}{\tilde{\mathbf{E}}_{xi}^3}\end{aligned} \tag{24}$$

is an auxiliary function for $\mathcal{J}_H(\mathbf{E})$. Here, $\tilde{\mathbf{\Theta}} = \mathbf{H}^T\{\mathbf{C}\circ[\mathbf{H}(\mathbf{U}\circ\tilde{\mathbf{E}})(\mathbf{U}\circ\tilde{\mathbf{E}})^T\mathbf{H}^T]\}\mathbf{H}(\mathbf{U}\circ\tilde{\mathbf{E}})$ and $\tilde{\mathbf{\Phi}} = \mathbf{H}^T(\mathbf{B}\circ\mathbf{C})\mathbf{H}(\mathbf{U}\circ\tilde{\mathbf{E}})$. Moreover, this function is a convex function in $\mathbf{E}$ and its global minimum is

$$\mathbf{E}_{xi} = \tilde{\mathbf{E}}_{xi}\left(\frac{4(\tilde{\mathbf{\Phi}}\circ\mathbf{U})_{xi}}{4(\tilde{\mathbf{\Theta}}\circ\mathbf{U})_{xi}+\tau}\right)^{\frac{1}{4}} \tag{25}$$

Briefly, Theorem 4 can be proved by validating $\mathcal{Z}_E(\mathbf{E},\tilde{\mathbf{E}}) \geq \mathcal{J}_H(\mathbf{E})$, $\mathcal{Z}_E(\mathbf{E},\mathbf{E}) = \mathcal{J}_H(\mathbf{E})$, and the Hessian matrix $\nabla^2_{\mathbf{E}}\mathcal{Z}_E(\mathbf{E},\tilde{\mathbf{E}}) \succeq \mathbf{0}$. The details of its proof can be found in the Supplementary Material.

From Definition 1 and Theorem 4 (Note Eq. (25) is consistent with Eq. (23)), at any iteration $\kappa \geq 1$, we have

$$\mathcal{J}_H(\mathbf{E}^{(\kappa)}) = \mathcal{Z}_E(\mathbf{E}^{(\kappa)},\mathbf{E}^{(\kappa)}) \geq \mathcal{Z}_E(\mathbf{E}^{(\kappa+1)},\mathbf{E}^{(\kappa)}) \geq \mathcal{J}_H(\mathbf{E}^{(\kappa+1)})$$

Thus $\mathcal{J}_H$ monotonically decreases. Since Eq. (16) is bounded below by zero, the updating of $\mathbf{E}$ will converge, and Theorem 3 is proved. Using a similar approach, we have proved Theorem 1 and 2. Thus the iterative algorithms of both Phase I and II will converge, which proves the convergence of Alg. 1.

Complexity Analysis. Let n and m be the number of nodes and edges in $\mathbf{A}$, respectively. The time complexity for updating $\mathbf{U}$ and $\mathbf{s}$ is $O(T_1(mk+nk^2))$, where T_1 is the number of iterations in Phase I. Let T_2 and T_3 be the number of iterations for updating $\hat{\mathbf{H}}$ and $\mathbf{E}$, respectively, then the time for updating $\hat{\mathbf{H}}$ is $O(T_2mn)$ using sparse matrix multiplication, the time for updating $\mathbf{E}$ is $O(T_3n^2k)$. Therefore, let $T_m = \max(T_1,T_2,T_3)$, the overall time complexity of CRD is $O(T_mn^2)$, considering k is a small constant and $\mathbf{A}$ (and $\tilde{\mathbf{A}}$) is often sparse s.t. m is linear w.r.t. n. In practice, we find Alg. 1 often converges fast, with a small T_m. Please see Sec. V-E for further details.

IV. Leveraging Multiple Temporal Broken Networks

As discussed in Sec. I, causal anomalies are usually stable in a short time period when fault propagates in the system [7]. Therefore, jointly analyzing multiple temporal broken networks has the potential to resolve the ambiguities in each snapshot that are brought by system noises. Next, we introduce how to extend CRD to a temporal setting.

First, in Phase I, we can replace $\mathbf{B}$ and $\mathbf{W}$ in $\mathcal{J}_B$ in Eq. (10) by $\mathbf{B}^{(t)}$ and $\mathbf{W}^{(t)}$ respectively to indicate the time point t. Let $\mathcal{J}_B^{(t)}$ represent the resulting function in Eq. (10) using $\mathbf{B}^{(t)}$ and $\mathbf{W}^{(t)}$, then $\mathcal{J}_{CR}$ in Eq. (11) becomes

$$\mathcal{J}_{CR}^{(t)}(\mathbf{U},\mathbf{s},\boldsymbol{\omega}) = \mathcal{J}_A + \beta\sum_{t=1}^{T}\boldsymbol{\omega}_t\mathcal{J}_B^{(t)} + \gamma\|\boldsymbol{\omega}\|_F^2 \tag{26}$$

where a weighting vector $\boldsymbol{\omega} = [\boldsymbol{\omega}_1,...,\boldsymbol{\omega}_T]$ is introduced to control the relative contributions of the successive broken networks in the time window T. By optimizing Eq. (26), inconsistent or non-informative snapshot $\mathbf{B}^{(t)}$ will learn a small $\boldsymbol{\omega}_t$ to reduce its negative impact. The ℓ_2 norm on $\boldsymbol{\omega}$ is used to avoid overfitting. γ is a regularization parameter. Typically, a large γ results in more non-zero entries in $\boldsymbol{\omega}$. Moreover, to learn interpretable weights, we also enforce constraints $\boldsymbol{\omega} \geq \mathbf{0}$ and $\sum_{t=1}^{T}\boldsymbol{\omega}_t = 1$.

Similarly, in Phase II, the objective function in Eq. (16) can be generalized to

$$\begin{aligned}\mathcal{J}_H^{(t)}(\mathbf{E},\hat{\boldsymbol{\omega}}) = &\sum_{t=1}^{T}\hat{\boldsymbol{\omega}}_t\|\mathbf{C}\circ[\mathbf{H}(\mathbf{U}\circ\mathbf{E})(\mathbf{U}\circ\mathbf{E})^{\mathrm{T}}\mathbf{H}^{\mathrm{T}}] - \mathbf{B}^{(t)}\|_F^2\\ &+\tau\|\mathbf{E}\|_1 + \gamma\|\hat{\boldsymbol{\omega}}\|_F^2\end{aligned} \tag{27}$$

where $\hat{\boldsymbol{\omega}}$ is another weighting vector. We also enforce $\hat{\boldsymbol{\omega}} \geq \mathbf{0}$ and $\sum_{t=1}^{T}\hat{\boldsymbol{\omega}}_t = 1$.

The optimization formula of $\mathbf{U}$, $\mathbf{s}$ and $\mathbf{E}$ for Eq. (26) and Eq. (27) are the same as before in Eq. (20), Eq. (21) and Eq. (23), except that in Eq. (20) and Eq. (21), $\mathbf{B}$ is replaced by $\sum_{t=1}^{T}\boldsymbol{\omega}_t\mathbf{B}^{(t)}$, $\mathbf{W}$ is replaced by $\sum_{t=1}^{T}\boldsymbol{\omega}_t\mathbf{W}^{(t)}$ and in Eq. (23), $\mathbf{B}$ is replaced by $\sum_{t=1}^{T}\hat{\boldsymbol{\omega}}_t\mathbf{B}^{(t)}$. In Eq. (26) (or Eq. (27)), the subproblem w.r.t. $\boldsymbol{\omega}$ (or $\hat{\boldsymbol{\omega}}$) is a convex problem and can be solved using existing solvers [18]. For brevity, we omit the details. In the following, we refer to the algorithm leveraging temporal broken networks as CRD-tmp.

V. Experimental Results

In this section, we conduct extensive experiments to evaluate the performance of CRD algorithm in real-world information system and cyber-physical system. The codes and datasets are available online[2].

A. Dataset Description

We evaluate the performance of CRD on two real-life large-scale system datasets: 1). bank information system dataset (BIS), and 2). power plant sensor dataset (PPS).

Bank Information Systems (BIS). The BIS dataset [2], [8] contains $1,273$ flow intensity time series monitoring different aspects of the system, such as CPU usage, disk I/O, etc., and each time series is collected every 6 seconds. The training data is collected at normal system states, where 168 time points are collected for each time series. The invariant network is then generated on the training data as described in Sec. II-A, which has $1,273$ nodes and $39,116$ edges. The testing data are collected during abnormal system states, where 169 time points are collected for each of the $1,273$ time series. As described in Sec. II-A, we use the testing data to track the changes of the invariant network, and generate broken networks. Among the 169 times points in the testing data, system experts observed a system failure at $t = 120$ and $t = 122$. Thus two broken networks on these time points are generated for performing anomaly ranking. There are

[2]https://www.dropbox.com/s/cxgr528ut9litzs/crd.zip?dl=0

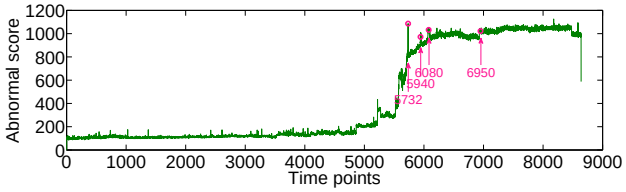


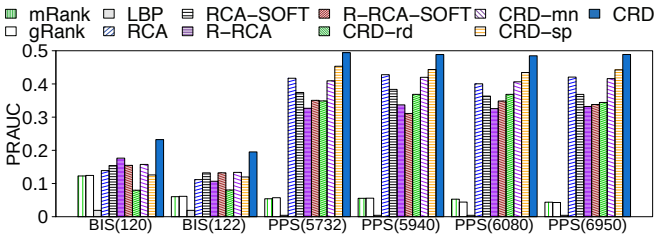
Fig. 3. Abnormal scores at different time points on PPS dataset.

$18,052$ and $16,089$ broken edges on $t = 120$ and $t = 122$, respectively. According to system experts, the root cause anomaly at these time points is related to "DB16-". Thus, we extract all time series with prefix "DB16-" in their titles as the ground truth anomalies. In total, there are 80 such time series. Our goal is to evaluate the capabilities of different methods to detect these causal anomalies.

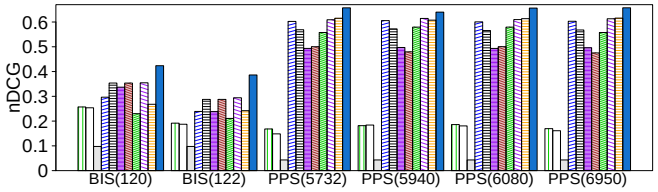
Power Plant Sensors (PPS). This dataset was collected by NEC Lab (www.nec-labs.com). It contains $6,049$ times series monitored by sensors distributed in a power plant system. Similar to the BIS data, the invariant network is trained using the time series collected in one normal day, where each time series is collected every 10 seconds and contains $8,639$ time points. The obtained invariant network contains $6,049$ nodes and $16,361$ edges. On another day when system failure happened, system experts generated an "abnormal score" for each collected time point, as shown in Fig. 3. This score is proportional to the total residual (Eq. (3)) at each time point. To detect causal anomalies, four time points with peak abnormal scores (as pinpointed in Fig. 3) are picked to generate four broken networks, which have 883, 930, $1,032$ and $1,022$ broken edges, respectively. Among them, $t = 5,940$ is the reported time when a system failure is observed by system operators. According to system experts, the problem is related to "XY2-" and "XY345-" sensors. Thus, we extract time series with prefix "XY2-" or "XY345-" in their titles as the ground truth. In total, there are 67 such time series.

B. Experimental Setup

The State-of-the-art Methods. We compare the performance of CRD with several state-of-the-art methods, including (1) mRank [2]; (2) gRank [2]; (3) LBP [8]; (4) RCA family algorithms [10]. The mRank and gRank methods rank causal anomalies mostly by the percentage of associated broken invariants with each node. The difference is that mRank only considers broken invariants directly linked to each node while gRank considers broken invariants within several hops to each node. LBP models a broken network by Markov Random Fields and infers causal anomalies using a loopy belief propagation algorithm. RCA family algorithms are based on label propagation but simply assume a single global propagation along a whole network. There are four variants, RCA, R-RCA, RCA-SOFT and R-RCA-SOFT. Here "R-" represents a relaxed version that runs faster than basic RCA. "-SOFT" means a softmax normalization is employed to avoid too large or too small ranking scores. We consider all these algorithms



(a) PRAUC comparison



(b) nDCG comparison

Fig. 4. Effectiveness evaluation on BIS and PPS datasets.

for comparison. The parameters of all methods are tuned to achieve their best performances.

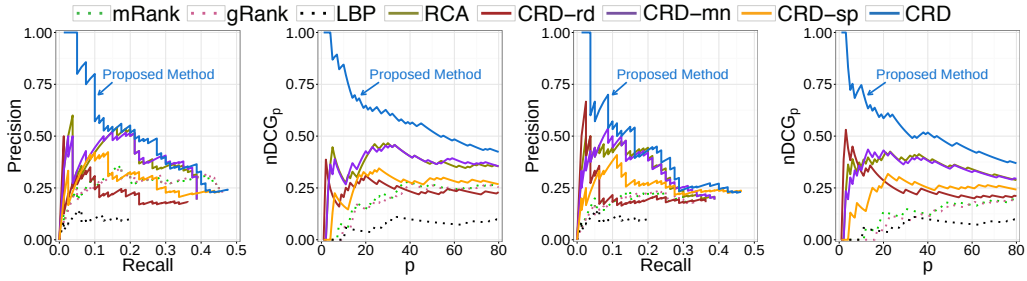
To study the effectiveness of each individual component of CRD, we also examine three degraded variants of CRD, i.e., CRD-rd, CRD-mn and CRD-sp. CRD-rd only uses Phase I where nodes in a cluster with high broken score $\mathbf{s}_i$ are ranked higher than those in a cluster with low $\mathbf{s}_i$. The nodes in the same cluster are randomly ordered. CRD-mn only uses Phase II by ignoring $\mathbf{U}$ in Eq. (16), and the anomaly score of a node x is $\mathbf{f}_x = mean(\mathbf{E}_{x*})$ without using $\mathbf{U}$ and $\mathbf{s}$. CRD-sp is the same as CRD except that in Phase I, it optimizes $\mathcal{J}_A$ and $\mathcal{J}_B$ separately (see Eq. (11)). Comparing with the first two variants can show the importance of integrating the two phases. Comparing with the third variant can show the effectiveness of joint optimization in Phase I.

Evaluation Criteria. Similar to existing works [2], [8], [10], we use area under precision-recall curve (PRAUC) [19] and normalized Discounted Cumulative Gain (nDCG) [20] to evaluate causal anomaly detection accuracy.

The precision-recall curve is calculated by varying the rank threshold from 1 up to K, where K is typically chosen as twice the actual number of ground truth causal anomalies [20], [8]. PRAUC has been considered to be better than AUC (area under the ROC curve) because PRAUC punishes highly ranked false positives much more than AUC does [19]. This is important in practice since only highly ranked anomalies will be examined by system experts in the diagnosis procedure. Thus highly ranked false positives should be avoided.

The nDCG at top-p ranking result is defined as $\text{nDCG}_p = \frac{\text{DCG}_p}{\text{IDCG}_p}$, where $\text{DCG}_p = \sum_{x=1}^{p} \frac{2^{\text{rel}_x}-1}{\log_2(1+x)}$ is defined on the inferred ranking list, and $\text{rel}_x = 1$ iff node x is a ground truth anomaly. IDCG_p is the DCG_p value on the ground truth ranking list. Here p is smaller than or equal to the number of ground truth causal anomalies.

For both evaluation measures, a larger value indicates a better performance.



(a) PR-curve ($t = 120$) (b) nDCG curve ($t = 120$) (c) PR-curve ($t = 122$) (d) nDCG curve ($t = 122$)

Fig. 5. The Precision-Recall curve and nDCG$_p$ curve on the BIS dataset.

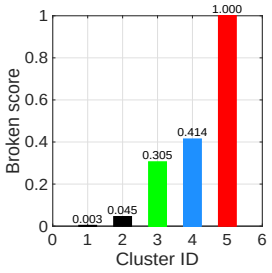


Fig. 6. The broken cluster scores.

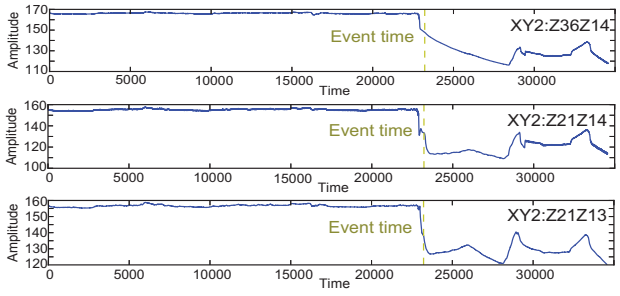
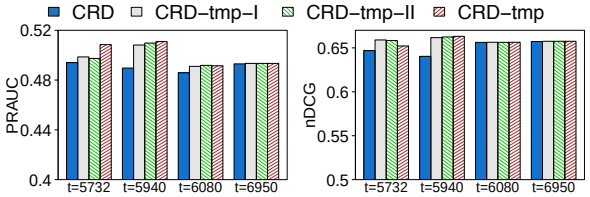


Fig. 7. Top 3 anomalies detected by CRD on PPS dataset.



(a) PRAUC comparison (b) nDCG comparison

Fig. 8. Results on leveraging temporal broken networks.

C. Effectiveness Evaluation

Fig. 4 shows the PRAUC and nDCG values of different methods on the BIS and PPS datasets. Here nDCG$_p$ is the value with p equal to the number of ground truth of each dataset. From the results, we have several key observations. First, CRD significantly outperforms other methods by large margins on all datasets. The PRAUC improvement over the best competing method on each dataset varies from 14.20% to 31.64%. This validates the importance of modeling local propagations of causal anomalies in their associated clusters. Moreover, by comparing CRD with CRD-rd and CRD-mn, we also see the importance of integrating Phases I and II as a coherent approach. Furthermore, the comparison with CRD-sp shows the effectiveness of joint optimization in Phase I of CRD. This also implies that an inferior clustering in Phase I can largely reduce the subsequent anomaly inference accuracy.

To better understand the difference between CRD and other methods, we have examined the Precision-Recall and nDCG$_p$ curves of different methods. For example, Fig. 5 presents these curves on BIS dataset. Here for clarity, we use R-RCA-SOFT to represent the RCA family since it generally performs better than other RCA algorithms on this data. From these results, we find the top ranked nodes by CRD contain more ground truth nodes than other methods, as demonstrated by the high heads of the curves of CRD. Hence CRD is practically more useful than other methods since system experts usually only check the top ranked nodes in a regular diagnosis.

We also perform a detailed evaluation of the broken cluster ranking component in Phase I of CRD. In Fig. 2(a) and (b), we have shown the invariant and broken networks for dataset BIS ($t = 122$). There are five clusters shown in the invariant network, 3 of which are broken and are highlighted in Fig. 2(b). Fig. 6 shows the broken scores learned by CRD for all detected clusters (i.e., $\mathbf{s}_i$ in Eq. (11)). The colors of the bars correspond to the clusters in Fig. 2(b). As we can see, the scores accurately reflect the broken degrees of different clusters. Furthermore, there is a clear difference between the scores of the broken clusters and unbroken clusters.

Finally, we present some case studies. Taking PPS dataset as an example, Fig. 7 shows the time series of the top 3 anomalous nodes detected by CRD at the time when system failure was observed (i.e., $t = 5940$). In the figure, the dashed line represents the system failure time. Obviously, the behaviors of all these time series become anomalous from the event time on, which validates the effectiveness of CRD.

D. Leveraging Temporal Broken Networks

In this section, we evaluate the capability of our method to leverage multiple temporal broken networks. In Sec. IV, we have proposed CRD-tmp to integrate successive broken networks in both Phase-I and II. To fully assess the effectiveness of our model, we denote CRD-tmp-I as the method only uses Phase I smoothing, i.e., Eq. (26), and let CRD-tmp-II be the method only uses Phase II smoothing, i.e., Eq. (27).

Fig. 8 shows the comparison between CRD and different smoothing strategies on the PPS datasets, where we set the window size $T = 10$ for temporal methods. From the results, we observe the temporal methods can effectively improve the accuracies at the first two time points, and CRD-tmp often outperforms other methods. This suggests that the smoothing strategies are useful in both phases and a combination is even better. At the last two time points, all methods perform

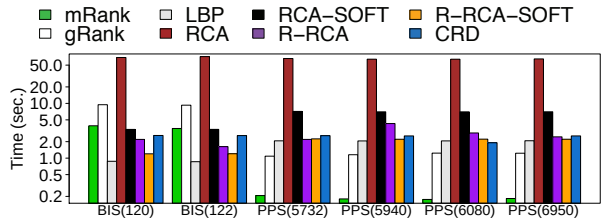
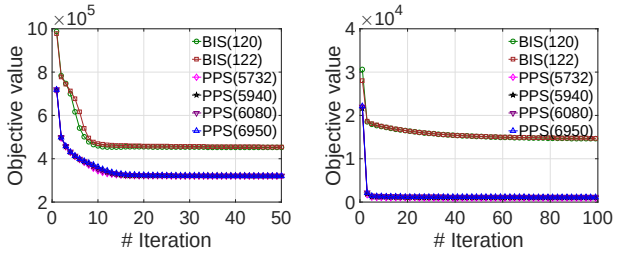


Fig. 9. Runtime of different methods on the BIS and PPS datasets.



(a) Phase-I (b) Phase-II

Fig. 10. Number of iterations to converge of CRD.

comparably. This is because at these time points, the status of the system are stable (see. Fig 3). Therefore, more broken networks will not provide more information for the temporal methods to leverage.

E. Runtime and Convergence Study

In this section, we study the efficiency of CRD in terms of its running time and the number of iterations for convergence. Fig. 9 shows the running time of the compared methods on the BIS and PPS datasets. We can see that CRD is comparable to those efficient methods. Note that BIS and PPS datasets represent large-scale system datasets. In practice, such large systems are typically equipped with thousands of sensors (i.e., nodes) [2]. CRD often runs 2 to 3 seconds on such data using a regular PC. Comparing with other methods, the high accuracy and the light overhead of running time suggest the practical value of CRD in real-world systems.

Next, Fig. 10(a) and 10(b) show the objective values of Phase I (Eq. (11)) and Phase II (Eq. (16)) w.r.t. the number of iterations on different datasets. As can be seen, both Phase I and II decrease objective values monotonically and often converge within 50 to 100 iterations. The fast convergence rate also demonstrates CRD is efficient in practice.

F. Parameter Sensitivity Study

There are three parameters in Phase I. α ($\alpha \geq 1$) is used to control the complexity of $\mathbf{U}$. β balances the two terms of the joint optimization in Eq. (11). k is the number of clusters. In Phase II, c balances the propagation and seed preference. τ controls the sparsity of results. Next, we will show that most of these parameters are easy to tune, either because of their small impacts or the existence of standard choices.

Taking BIS (120) dataset as an example, Fig. 11 shows the impacts of each parameter on the inference accuracy. By default, we set $\alpha = 1.1$, $\beta = 0.2$, $k = 7$, $c = 0.8$ and $\tau = 1$. From the figures, the performance is stable w.r.t. α and τ in wide ranges. c can be set by the typical values used in random walk, which is around 0.8 [21]. k shows similar impacts in a relatively wide range (note we usually do not set $k = 1$). β should be tuned with some efforts. As can be seen, usually a small β is preferred. The non-zero choices of β also suggests the importance of joint optimization in Eq. (11).

VI. Related Work

Related work can be grouped into two categories. In this section, we first discuss the work on fault detection in distributed systems. Then we discuss graph (or network)-based anomaly detection methods.

Fault detection has been studied in many different fields [6]. In information systems, Yemini et al. [22] proposed to locate system faults based on known dependency relationships between faults and symptoms, but in practice, how to obtain such fault-symptom dependency relationships precisely remains a difficult issue. In circuit and software diagnosis, model-based approach [23] and spectrum-based approach [24] are two major fault detection methods. These methods are useful on logical data, but are not designed for analyzing time series data. For time series data, the nearest neighbor based methods [25] and PCA based methods [26] were proposed to locate anomalous patterns. However, they are not designed to find causal anomalies, which are most important for system debugging and recovery.

Recently, there is a growing attention on graph-based anomaly detection methods, using either static or dynamic graphs [6], [27]. In static graphs, the main task is to spot anomalous network entities, including nodes, edges, and subgraphs. For example, in [28], OddBall was designed to detect anomalous nodes in weighted graphs. In [29], frequent subgraph mining has been used to detect non-crashing bugs in software-flow graphs. More recently, methods to detect anomalous nodes in attributed graphs have also been studied [30], [31]. However, these approaches only focus on a single graph. In comparison, we consider both the invariant and broken graphs, which provide a more dynamic and complete picture for anomaly detection. In dynamic graphs, many existing methods were proposed to detect abnormal events, and the time stamps when those events take place [32]. In [33], a tensor reconstruction method was proposed to detect anomalous entities having large reconstruction errors in dynamic graphs. These methods, however, are not designed to rank system components for locating causal anomalies.

Some methods use "correlation graph" for anomaly detection [34], [35], [36]. In a correlation graph, nodes represent sensors, and each edge is weighted by the correlation coefficient between a pair of time series recordings on the sensors. These methods aim to identify nodes whose associated edge weights change a lot in a certain period. In this way, nodes with anomalous behaviors can be obtained. However, which nodes are causal anomalies still remains unknown.

There are many methods using invariant graphs constructed from the monitoring data for system analysis [5], [7], [1], [9].

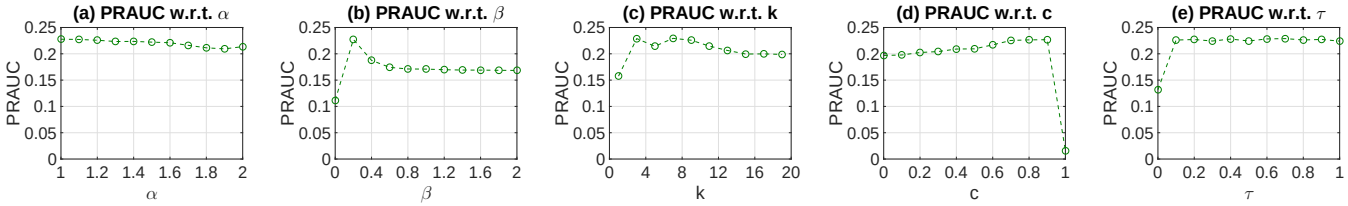


Fig. 11. Parameter study on BIS dataset. Each time, one parameter is varied when others are fixed.

In [2], mRank and gRank were proposed to detect anomalies in an invariant network. However, these methods ignore the fault propagation and heavily rely on the percentage of broken invariants within the neighborhood of each node. In [8] and [10], two different ways were proposed to model fault propagation, both of which assume a single global propagation in the whole network. As has been discussed, this is not precise in many emerging applications. In fact, multiple local propagations happening in different clusters can jointly define a system fault. Such global methods are not aware of this scenario, making them sub-optimal in locating causal anomalies.

VII. Conclusion

Automatically detecting causal anomalies is a crucial task in system management. The state-of-the-art methods for causal anomaly detection assume a single and global fault propagation in the invariant network, which may not be true in real-world complex systems. In this paper, we propose a novel algorithm CRD to model multiple and local fault propagations in different clusters of the invariant network. CRD first finds broken clusters and ranks them by their broken degrees. Then it backtracks causal anomalies in different clusters using a principled low-rank network diffusion model. Experimental results on real-life datasets demonstrate CRD consistently outperforms the competitors.

References

[1] H. Chen, H. Cheng, G. Jiang, and K. Yoshihira, "Exploiting local and global invariants for the management of large scale information systems," in *ICDM*, 2008.

[2] Y. Ge, G. Jiang, M. Ding, and H. Xiong, "Ranking metric anomaly in invariant networks," *ACM Trans. Knowl. Discov. Data.*, vol. 8, no. 2, p. 8, 2014.

[3] D. Djurdjanovic, J. Lee, and J. Ni, "Watchdog agent-an infotronics-based prognostics approach for product performance degradation assessment and prediction," *Adv. Eng. Inform.*, vol. 17, no. 3, pp. 109–125, 2003.

[4] J. Gertler, *Fault detection and diagnosis in engineering systems*. Marcel Dekker, 1998.

[5] G. Jiang, H. Chen, and K. Yoshihira, "Discovering likely invariants of distributed transaction systems for autonomic system management," in *ICAC*, 2006.

[6] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Comput. Surv.*, vol. 41, no. 3, p. 15, 2009.

[7] G. Jiang, H. Chen, and K. Yoshihira, "Modeling and tracking of transaction flow dynamics for fault detection in complex systems," *IEEE Trans. Dependable Secure Comput.*, vol. 3, no. 4, pp. 312–326, 2006.

[8] C. Tao, Y. Ge, Q. Song, Y. Ge, and O. A. Omitaomu, "Metric ranking of invariant networks with belief propagation," in *ICDM*, 2014.

[9] M. Momtazpour, J. Zhang, S. Rahman, R. Sharma, and N. Ramakrishnan, "Analyzing invariants in cyber-physical systems using latent factor regression," in *KDD*, 2015.

[10] W. Cheng, K. Zhang, H. Chen, G. Jiang, and W. Wang, "Ranking causal anomalies via temporal and dynamical analysis on vanishing correlations," in *KDD*, 2016.

[11] L. Ljung, *System identification: theory for the user, 2nd ed*. Prentice Hall PTR, 1999.

[12] G. Jiang, H. Chen, and K. Yoshihira, "Efficient and scalable algorithms for inferring likely invariants in distributed systems," *IEEE Trans. Knowl. Data Eng.*, vol. 19, no. 11, pp. 1508–1523, 2007.

[13] Z. Yang and E. Oja, "Clustering by low-rank doubly stochastic matrix decomposition," in *ICML*, 2012.

[14] J. Yang, J. McAuley, and J. Leskovec, "Community detection in networks with node attributes," in *ICDM*, 2013, pp. 1151–1156.

[15] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf, "Learning with local and global consistency," in *NIPS*, 2004.

[16] D. D. Lee and H. S. Seung, "Algorithms for non-negative matrix factorization," in *NIPS*, 2001.

[17] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[18] G. Yu, H. Rangwala, C. Domeniconi, G. Zhang, and Z. Zhang, "Protein function prediction by integrating multiple kernels," in *IJCAI*, 2013.

[19] J. Davis and M. Goadrich, "The relationship between precision-recall and roc curves," in *ICML*, 2006.

[20] K. Järvelin and J. Kekäläinen, "Cumulated gain-based evaluation of ir techniques," *ACM Trans. Inf. Syst.*, vol. 20, no. 4, pp. 422–446, 2002.

[21] T. Haveliwala and S. Kamvar, "The second eigenvalue of the google matrix," Stanford, Tech. Rep., 2003.

[22] S. A. Yemini, S. Kliger, E. Mozes, Y. Yemini, and D. Ohsie, "High speed and robust event correlation," *IEEE communications Magazine*, vol. 34, no. 5, pp. 82–90, 1996.

[23] J. De Kleer and B. C. Williams, "Diagnosing multiple faults," *Artificial intelligence*, vol. 32, no. 1, pp. 97–130, 1987.

[24] R. Abreu, P. Zoeteweij, and A. J. Van Gemund, "Spectrum-based multiple fault localization," in *ASE*, 2009.

[25] E. Keogh, J. Lin, and A. Fu, "Hot sax: Efficiently finding the most unusual time series subsequence," in *ICDM*, 2005.

[26] R. Jiang, H. Fei, and J. Huan, "Anomaly localization for network data streams with graph joint sparse pca," in *KDD*, 2011.

[27] L. Akoglu, H. Tong, and D. Koutra, "Graph based anomaly detection and description: a survey," *Data Min. Knowl. Discov.*, vol. 29, no. 3, pp. 626–688, 2015.

[28] L. Akoglu, M. McGlohon, and C. Faloutsos, "Oddball: Spotting anomalies in weighted graphs," in *PAKDD*, 2010.

[29] C. Liu, X. Yan, H. Yu, J. Han, and S. Y. Philip, "Mining behavior graphs for "backtrace" of noncrashing bugs." in *SDM*, 2005.

[30] J. Gao, F. Liang, W. Fan, C. Wang, Y. Sun, and J. Han, "On community outliers and their efficient detection in information networks," in *KDD*, 2010.

[31] B. Perozzi, L. Akoglu, P. Iglesias Sánchez, and E. Müller, "Focused clustering and outlier detection in large attributed graphs," in *KDD*, 2014.

[32] R. A. Rossi, B. Gallagher, J. Neville, and K. Henderson, "Modeling dynamic behavior in large evolving graphs," in *WSDM*, 2013.

[33] J. Sun, D. Tao, and C. Faloutsos, "Beyond streams and graphs: dynamic tensor analysis," in *KDD*, 2006.

[34] T. Idé, S. Papadimitriou, and M. Vlachos, "Computing correlation anomaly scores using stochastic nearest neighbors," in *ICDM*, 2007.

[35] T. Idé, A. C. Lozano, N. Abe, and Y. Liu, "Proximity-based anomaly detection using sparse structure learning," in *SDM*, 2009.

[36] S. Hara, T. Morimura, T. Takahashi, H. Yanagisawa, and T. Suzuki, "A consistent method for graph based anomaly localization," in *AISTATS*, 2015.