Effective Attribute Selection for Multi-dimensional Root Cause Analysis

Yiran Cheng†, Bo Cheng§, Pengxiang Jin‡, Yongqian Sun*‡, Xiaohui Nie§, Nengwen Zhao†, Shenglin Zhang‡, Dan Pei†
†Tsinghua University, {chengyr20, znw17}@mails.tsinghua.edu.cn, peidan@tsinghua.edu.cn,
‡Nankai University, jinpengxiang@mail.nankai.edu.cn, {sunyongqian, zhangsl}@nankai.edu.cn,
§BizSeer, cb229435444@gmail.com, niexiaohui@bizseer.com

***Abstract*—Using large-scale multi-dimensional data for root cause analysis (MDRCA) is vitally important for online software services. It helps operators narrow down the scope of anomalies and failures quickly and localize the root cause to a finer granularity. However, most existing MDRCA algorithms can only solve low-dimensional problems. When dealing with high-dimensional data, the complexity of these algorithms would significantly increase, and even some algorithms would no longer work. Intuitively, passing only a subset of attributes rather than full attributes can improve the performance of these MDRCA algorithms. However, it is challenging due to data imbalance and novel root cause attributes. To better understand the problem of root-cause-oriented attribute selection (RCOAS), we conduct a preliminary study based on real-world data. We find that there exist several straightforward rules to filter out some attributes. In addition, we reveal that existing approaches do not fit the requirements of RCOAS. Motivated by the study, we propose an RCOAS approach, RC-LIR, to select a subset of attributes for downstream algorithms. RC-LIR first performs rule-based selection. Then it improves a feature selection algorithm by two strategies, i.e., scaling up imbalanced data and considering the redundant cost. Experiments on 1000 real-world fault cases demonstrate that RC-LIR can achieve an F1-score of 0.88, outperforming the baseline approaches by at least 0.15. Furthermore, our experiments with four widely adopted MDRCA algorithms show that integrating RC-LIR can lead to more effective and efficient MDRCA.**

***Index Terms*—Multi-dimensional Data, Attribute Selection, Root Cause Analysis, Logistic Iterative Relief**

I. Introduction

MULTI-DIMENSIONAL data is prevalent in online software services, and it is widely used for Root Cause Analysis (RCA) [1], [2], [3], [4]. Generally, operators obtain multi-dimensional data by instrumenting agents that continuously record multi-dimensional logs. For example, Table I shows several access logs of an online software service. Each log contains a timestamp, a key performance indicator (KPI) (i.e., *is_success*) and 6 attributes, i.e., *user_id, user_name, request_type, src, dst, home_page*. The KPI is the main concern of operators, for it usually reflects the user experience and service provider's revenue. When the KPI suffers from an abnormal increase or decrease, operators need to localize the root cause timely. Typically, operators aggregate the raw logs to get formal multi-dimensional data. Table II lists the aggregated multi-dimensional data. Logs with the same *timestamp, request_type, src,* and *dst* are grouped in this case. In the aggregated table, we can easily find the anomalously low KPI values (i.e., 1/80, 1/154 and 1/113), and the corresponding root cause is a two-attribute combination, i.e., (*request_type* $= 1$, *src* $=$ *host2*).

Over the years, researchers have proposed many automatic multi-dimensional root cause analysis (MDRCA) approaches, e.g., Apriori [5], [6], iDice [2], Hotspot [3], and Squeeze [4]. Typically, the root cause localized by these algorithms is

TABLE I
Access logs as a example of multi-dimensional data

time-stamp	*user _id*	*user _name*	*request _type*	*src*	*dst*	*home _page*	is_ success
16251 04800	1203	alice123	0	host0	server1	example .com	1
16251 04800	1450	booob	1	host2	server0	example .com	1
16251 06600	51	carol95	0	host0	server1	example .com	0
16251 06600	1450	booob	2	host0	server1	example .com	1
16251 08400	51	carol95	0	host0	server1	example .com	0
16251 08400	1203	alice123	0	host3	server2	example .com	1
16251 08400	1450	booob	1	host1	server3	example .com	1

TABLE II
Example of aggregated multi-dimensional data

timestamp	*request_type*	*src*	*dst*	success_ratio (#success/#request)
1499961600	0	host1	server0	120/120
1499961600	**1**	**host2**	server3	**1/80**
1499961600	0	host1	server2	141/141
1499961600	0	host1	server1	302/305
1499961600	**1**	**host2**	server2	**5/154**
1499961600	2	host0	server2	129/130
1499961600	0	host2	server1	132/132
1499961600	2	host1	server2	147/149
1499961600	**1**	**host2**	server1	**2/113**

* Yongqian Sun is the corresponding author.

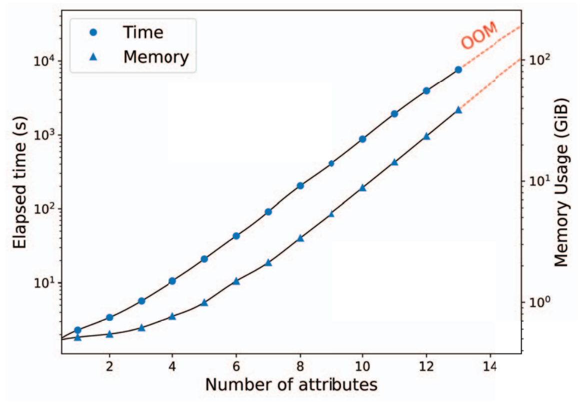

Fig. 1. The time and memory consumption of Squeeze, one of the state-of-the-art MDRCA algorithms, as the number of attributes increases

a combination of certain attributes and values. In the case of Table II, the root cause attributes are *request_type* and *src*, and the values are 1 and *host2*, respectively. There can be coarse-grained root causes (i.e., a combination of fewer attributes) and fine-grained root causes (i.e., a combination of more attributes). These approaches are proposed under varied architectures, and target root causes with different granularities. Operators can choose the MDRCA algorithm according to the need of their circumstances and their own favors. We will introduce some typical multi-dimensional RCA approaches in Section II-A.

However, most existing MDRCA algorithms can only work under low-dimensional conditions (usually less than 10 dimensions). The *efficiency* and *effectiveness* of these algorithms degrade dramatically when the number of dimensions becomes large. On the one hand, the computational consumption grows significantly, making these algorithms *inefficient*. For example, Fig. 1 shows the time and memory consumption when applying Squeeze [4] to conduct MDRCA with a different number of dimensions. We used a 32-core CPU and 256GB of RAM to run Fig.1. Please note that the Y-axis of Fig.1 is logarithmic. We can see that both the elapsed time and memory usage increase rapidly. Here, adding ten attributes will need 146x of computing power, while more attributes are also common in production environments. The situation worsens when there are over 13 attributes, for Squeeze runs into a crash due to out-of-memory (OOM), even with 64GiB of memory available. Other MDRCA algorithms face a similar situation. The reason is that the analysis of multi-dimensional data often needs to enumerate subsets of attributes, the number of which increases exponentially (m attributes results in $2^m - 2$ valid subsets). On the other hand, the large search space reduces the probability that the true root cause can be found, thus affecting the *effectiveness* of MDRCA algorithms.

The curse of dimensionality lies in the nature of multi-dimensional data analysis, thereby undermining a wide range of MDRCA algorithms. **Instead of proposing a new MDRCA algorithm, we attempt to propose a solution that can benefit a variety of existing MDRCA algorithms.** We use the term RCOAS (Root-Cause-Oriented Attribute Selection) to describe the task. Specifically, given the raw multi-dimensional data, our task is to filter out some attributes and then pass the filtered data to downstream MDRCA algorithms. *Downstream* means we do not modify any existing MDRCA algorithms, thus keeping their applicability. By filtering out root-cause-irrelevant attributes, we ease the problem of too many dimensions without loss of the accuracy of downstream algorithms. Naturally, feature selection (FS), which rules out redundant or irrelevant features in machine learning, comes into our sight.

However, applying FS to select attributes for MDRCA algorithms faces the following challenges (more details in Section II-C):

1) **Native FS does not fit RCOAS**. Native FS either cannot work on categorical data, which is prevalent in RCOAS, or does not work well on RCOAS because the target is different.
2) **Data imbalance**. RCOAS faces imbalanced data (i.e., #success cases are much more than #failure cases), while FS usually works on balanced data.
3) **Novel root cause attributes**. An attribute can be the root cause attribute for a new fault, even if it has never been a root cause attribute for any historical faults.

To tackle the challenges above, we propose an RCOAS approach in this paper, named RC-LIR: (i) based on an improved LIR algorithm [7], RC-LIR could handle the categorical values perfectly; (ii) RC-LIR uses an augmentation coefficient in LIR to sovlve the data imbalance problem; (iv) RC-LIR involves an redundant cost to consider the influence of no-fault values.

The contributions of this paper are summarized as follows:

1) To the best of our knowledge, we are among the first to identify the RCOAS problem. Precise RCOAS can reduce the computation consumption of MDRCA as well as improve their effectiveness in finding root causes. We formally define the problem, describe its difference from feature selection, and state its relationship to MDRCA.
2) We propose an RCOAS approach RC-LIR. RC-LIR first applies rule-based selecting to discard obviously redundant attributes. Moreover, RC-LIR improves a feature selection algorithm LIR by three modifications: defining categorical distance for challenge 1, partially scaling up origin data for challenge 2, and adding redundant cost for challenge 3.
3) We evaluate the performance of RC-LIR and downstream MDRCA algorithms using large-scale multi-dimensional data from a real-world enterprise. The results show that RC-LIR achieves an F1-score of 0.88, outperforming the baseline algorithms by 0.15, and it does improve the effectiveness and efficiency of downstream RCA algorithms a lot.

II. Background and Motivation

A. Necessity of Attribute Selection

Root cause analysis algorithms need to analyze the occurred fault and localize them to a set of attribute-value pairs to help recover from the fault. For example, in Table II, we can localize the fault to *request_type*=1 & *src*=*host2* (i.e.,

TABLE III
AN EXAMPLE OF FAULT CASE

	multi-dimensional attributes									KPI
	timestamp	*user_id*	*user_name*	*request_type*	*src*	*dst*	*home_page*	*digest*	…	is_success
N records	1625106000	1203	alice123	**1**	**host2**	server2	example.com	92ccdd	…	0
	1625106000	1450	bob98	1	host1	server3	example.com	8ecc9b	…	1
	1625106060	51	carol95	0	host2	server1	example.com	2e7812	…	1
	1625106060	1450	bob98	2	host3	server2	example.com	66aacc	…	1
	1625106120	51	carol95	**1**	**host2**	server1	example.com	2a6d14	…	0
	1625106120	1203	alice123	2	host3	server2	example.com	4ada68	…	1
	…	…	…	…	…	…	…	…	…	…
	1625106120	1450	bob98	**1**	**host2**	server3	example.com	c233a2	…	0
	Rule4	Rule3	Rule3	root cause attributes	root cause attributes		Rule2	Rule1		

the shaded cells). MDRCA algorithms usually drill down or enumerate data to check the potential contribution of each attribute-value pair to the fault and make the selection.

We can find that the above algorithms need to search results in a multi-dimensional space of subsets of attributes, which reduces the efficiency and accuracy in the case of high-dimension. Although some algorithms use pruning [2] and other strategies [8], it is still difficult to eliminate the impact (e.g., the efficiency of Squeeze in Fig. 1). Using all attributes of the multi-dimensional data as input [3], [4] without any filter, as we mentioned in Section I, could lead to the curse of dimensionality for some MDRCA algorithms, which will cause the elapsed time of the algorithm to rise to an unacceptable level. Another drawback is that many downstream algorithms [1], [2] will be affected by redundant attributes, which reduces the effect of analysis. Some RCA algorithms [4] may even crash when there are too many attributes.

B. *Definition of Attribute Selection*

We first introduce some important terms concerning an RCOAS (Root-Cause-Oriented Attribute Selection) algorithm. Then we give the formal definition of RCOAS.

Usage. As discussed in Section II-A, directly applying MDRCA algorithms faces the problem of too many attributes. Therefore, operators need to apply RCOAS to identify the critical attributes of a system periodically. When a new fault case is alerted, the operators only pass the critical attributes to MDRCA algorithms to determine the actual root cause. To retain the probability of finding the true root cause, we want to keep the possible root-cause-related attributes as well as filter out root-cause-irrelevant or redundant attributes when conducting RCOAS.

Fault case and **system**. A fault case can be regarded as a period of time where the KPI is degraded. Operators diagnose an online fault case by analyzing the data during the fault case. The fact that fault cases are much less than data used in ML cripples most common supervised models (more details in Section II-C).

Fig. 2 shows four example fault cases from the same system. In practice, the root cause attributes and values of fault cases can be different, often subject to the internal characteristics of

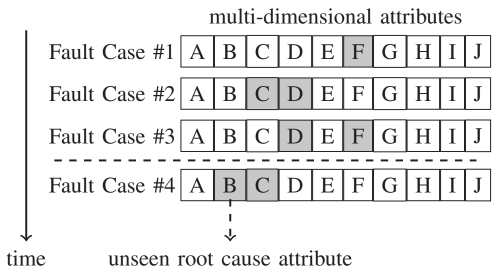

Fig. 2. The root cause attributes of four independent fault cases from the same systems. Each cell represents an attribute, and the shaded cell means the root cause attributes of the fault case. Only after the happening of fault case #4 and a thorough investigation can we can know that the attribute B can be root cause attribute. In other words, summarizing history fault cases is insufficient for RCOAS.

a system. Even if two systems share the same definition and collection of attributes, the probabilities of the same attribute being a root cause can be different.

The problem of RCOAS concerns a *system* rather than a single *fault case*, because there can be an unseen root cause attribute for a newly alerted fault case (Fault Case #4 in Fig. 2). In contrast, MDRCA only cares about the exact root cause localization for a specific *fault case*. We believe that combining RCOAS and MDRCA brings more effectiveness and efficiency, motivating us to propose an RCOAS solution.

Multi-dimensional attribute data. N denotes the number of total records. As shown in Table III, there are the attribute part and the KPI part of a record. For the KPI part, we treat failure indicating KPI as $\mathrm{positive}$ (e.g., 0 in Table III), and normal status as $\mathrm{negative}$ (e.g., 1 in Table III).

Root cause labeling v.s. KPI. The root cause is defined at the level of fault case, i.e., one fault case corresponds to one root cause (yet the root cause can be composed of several attributes and values). KPI is observed at the level of record, i.e., one record corresponds to one component of KPI. Note that **root cause labeling is different from KPI**. Supervised FS models utilize the KPI, but not a root cause, thus degrading their applicability for RCOAS, i.e., they try to predict the KPI but not the root cause. Moreover, we cannot train a supervised model on root cause labeling due to the high overhead of manual labeling (more details in Section II-C).

Definition of RCOAS. Formally, given a fault case y, and $\mathcal{A}$ as the set of all attributes of the raw multi-dimensional data, the goal of RCOAS is using the data of y to infer $\mathcal{S}$:

$$\max_{\mathcal{S}} |\mathcal{S} \cap \mathcal{R}| \, \text{s.t.} \, (\mathcal{S} \subsetneq \mathcal{A}) \wedge (|\mathcal{S}| \ll |\mathcal{A}|) \, , \qquad (1)$$

so that we can run MDRCA on $\mathcal{S}$, a subset of attributes. $\mathcal{R}$ is the set of critical attributes (possible root cause attributes) of the system where y takes place. $\cap$ is the union operation on sets. $\wedge$ is the "logical and" operation.

C. Preliminary Study and Challenges

In brief, attribute selection is about discarding and keeping attributes. But determining whether to discard or keep an attribute is not easy. We conduct an empirical study using 1000 fault cases from a real-world production environment. There are 45 attributes, and the critical attributes of these fault cases are labeled by experienced operators (for more details, see Section IV-A1).

We find that there are always attributes in the raw multi-dimensional data that will never become a root cause and should be filtered out. Next, we have three observations on the real-world multi-dimensional data that show attribute redundancy.

1) There are some columns that have great information gain but are not of analytical value and are just noise during maintenance. A simple example is the column *access_digest* in Table III. These attributes of any two records are unlikely to be identical, which means that the algorithm cannot classify the two attributes. More importantly, it is difficult for both operators and algorithms to derive valuable information from these two attributes (the *id* may imply some sort of chronological order, but we already have a *timestamp*). Therefore, we can think of these two attributes as redundant for the RCA process.
2) In Table III, the value of column *home_page* is always *example.com*, which means it does not contain any information gain and is, therefore, a redundant attribute.
3) Similarly, in the given table, the values of column *user_id* and *username* correspond one to one, such as a user with the *user_id* of *103* is *alice123*. Therefore, when analyzing multi-dimensional data, we can simply ignore one of the attributes, such as *user_id*, only need to analyze username, which makes us think *user_id* is redundant as an attribute.

The above observations give our straightforward rules to discard some attributes, but **determining what attributes to keep is far more difficult due to the highly heuristic nature of root cause analysis**. As we discussed in Section I, we can introduce feature selection methods to the multi-dimensional data attribute selection. One of the popular techniques in FS is to compute KL Divergence on features (i.e., attributes) and target variables (i.e., KPI). Fig. 3 shows the ineffectiveness of selecting attributes entirely up to KL divergence. If we select attributes to an acceptable number for downstream algorithms (e.g., less than 10 attributes), we can only cover 0.451% of the cases, thus missing the root-cause-attributes of a large number of fault cases (point A of Fig. 3); if we want to cover more than

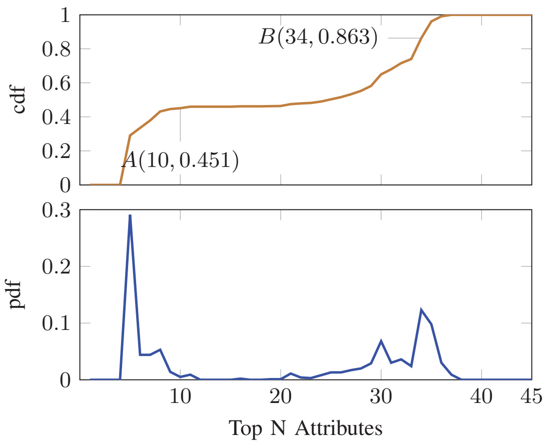

Fig. 3. Fault case coverage using popular feature selection technique. We compute the KL divergence of each attribute and select the top N attributes for 1000 real-world fault cases.

80% of the fault cases, the number of attributes will be too many (e.g., more than 34 attributes) for downstream algorithms to consume (point B of Fig. 3).

Besides the KL Divergence we discussed above, there are many more complex and accurate FS methods. However, as we will mention in the challenge 1 below, most of them are unsuitable for RCOAS. Specifically, multi-dimensional data poses the following additional challenges to attribute selection:

Challenge 1: **Native FS methods are not suitable for RCOAS.** We need to carefully select the FS methods and modify them to adapt to RCOAS. The defects of existing FS methods (which can be mainly divided into filter methods, embedded methods, and wrapper methods) are as follows:

a) Many filter FS methods, such as the Laplacian Score [9] and Relief [7], can not handle categorical values (values are ones of different types, e.g., *user_id* in Table III) in the multi-dimensional data since they need to calculate the mean, variance, norm, distribution, etc., which are not natively available for the categorical value. For example, you cannot calculate the mean value of *user_id* in Table III. The KL Divergence is one of the few filter methods that are suitable for categorical values, but it performs poorly according to our evaluation. In our approach, we make some modifications to the Relief-based algorithm so that its calculation can be applied to categorical values.
b) Some FS methods, especially embedded methods [10], [11], [12] and wrapper methods [13], [14], will call a specific ML algorithm to fit KPI by a subset of features, then evaluate the score of the feature subset by the fitting degree. However, for the RCOAS problem, our goal is to help the MDRCA to localize the root cause but not to fit the KPI, and the root cause and the KPI do not have one-to-one correspondence. This inconsistency leads to worse results of the FS methods.
c) For those embedded methods and wrapper methods like SVM, it is neither feasible to call an MDRCA algorithm and fit the root causes instead of the KPI because evaluating the MDRCA algorithm will need manually root cause labeling,

which is not easily available in our problem.

d) Typical non-FS dimensionality reduction algorithms (e.g., Factor analysis, discriminant analysis, PCA) will generate new dimensions that change the attributes. However, our requirement for RCOAS is to reduce the dimension of attributes for RCA, but cannot change the meaning of attributes, and we cannot accept the transformation of attributes.

Challenge 2: **Data imbalance.** As we will discuss in Section VI, most filter FS methods [7], [9] treat the problem as a binary or multi-classification problem, relying on fault labels. Here, fault labels mean KPI (e.g., *is_success* in Table III) but no root cause labels. However, there is always the data imbalance problem on fault labels due to faults being very rare among the whole dataset (in our dataset, $1:34.71$), which will be really challenging [15] for the FS methods.

Challenge 3: **Faults in the training dataset cannot cover all the possible root cause attributes.** Most existing supervised FS methods [13], [14], [16] will assume that the critical attributes and their values (for MDRCA, root causes) are independent and identically distributed among the training dataset and the test dataset, and the training dataset can fully cover the typical cases in the test dataset. However, in RCOAS, faults in the training dataset cannot cover all the possible root cause attributes due to the existence of 0-day faults (faults that never happened before) and rare faults. For example, as we discussed in Fig. 2 before, root cause B is an unseen root cause in the training dataset. Therefore, the FS method that only learns from faults will not be able to make a comparison between A and B, while A is a redundant attribute for MDRCA, but B is not. In our approach, we will try to remove A by comparing the redundancy of the attributes.

III. Approach

A. Overview of RC-LIR

In this paper, we propose an approach RC-LIR, an improved FS algorithm, to solve the RCOAS problem. Fig. 4 shows the overall framework of RC-LIR. At first, we apply a rule-based selecting process before the core RC-LIR approach, which can significantly speed up the approach and reduce false positives. For challenge 1, RC-LIR adopts a relief-based FS algorithm LIR, and improves it by redefining the distance function to accommodate the categorical values. Then, to tackle the data imbalance problem of challenge 2, RC-LIR **scale up** the influence of positive samples by adding an augmentation coefficient. In addition, RC-LIR consider the **redundant cost** of no-fault values while logistic regression to overcome challenge 3.

B. Rule-Based Selecting

Before starting the core part of the algorithm, we will remove the obvious redundant attributes according to the rules described in Section II-C.

1) According to the first rule, we first remove the attributes that have no analysis value due to all values being too unique. If all values of an attribute are strictly unique, we will remove this attribute. Similarly, if all values of an attribute do not have enough support for analysis, we will remove this attribute. In our algorithm, we set MIN_SUPPORT to $\max(1/N, 0.001)$, where N is the number of records, and all the attributes that fail to meet MIN_SUPPORT will be removed.
2) After that, if all values of an attribute are the same, it will also be removed.
3) Finally, after the first two rules are applied, if the information gain of an attribute relative to another attribute is 0, it will also be removed. In particular, if two attributes are redundant to each other, only one of them should be removed.
4) The algorithm will find a representative data timestamp attribute and discard it in the following steps because we think that the timestamp is indispensable for most algorithms, so we always keep the timestamp attribute.

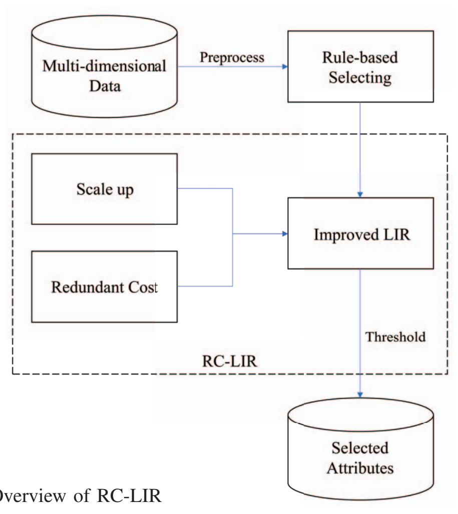

Fig. 4. Overview of RC-LIR

The bottom of Table III illustrates the application of the four rules. Through the automatic and quickly rule-Based selecting process, we reduce the running time of RC-LIR. Our experiments show that this rule-based selecting process reduces the average running time of the whole algorithm from 32.2 seconds to 24.6 seconds.

C. Improved LIR

Facing challenge 1 in Section II-C, we chose LIR as the base algorithm of RC-LIR due to the following reasons: i) we can make RC-LIR applicable to multi-dimensional data by defining the distance function of multi-dimensional data. ii) Relief-based algorithms do not rely on a specific ML algorithm to evaluate the weights of features.

Relief algorithm: Relief [7] is the base algorithm of LIR and our RC-LIR algorithm. At every training epoch of Relief, the algorithm randomly picks x from the dataset. Then, the algorithm searches for its two nearest neighbors: one from the same class (called nearest hit, $\mathrm{NH(x)}$) and the other from the other class (called nearest miss, $\mathrm{NM(x)}$). By $\mathrm{NH}(x)$ and $\mathrm{NM}(x)$, the weight w could be updated by

$$w = w + |x - \mathrm{NM}(x)| - |x - \mathrm{NH}(x)| \tag{2}$$

Repeatedly update w, then the algorithm can tell the final result.

I-Relief: I-Relief (Iterative RELIEF) [17] is an extension of Relief. I-Relief estimates w by using gradient descent to reach (3) [18].

$$\begin{aligned} &\min_w \sum_n w^T z_n \\ &\text{s.t. } \|w\|_2^2 = 1,\ w \geq 0 \\ &\text{where } z_n = d\left(x_n, \mathrm{NM}\left(x_n\right)|w\right) - d\left(x_n, \mathrm{NH}\left(x_n\right)|w\right) \end{aligned} \tag{3}$$

For RCOAS, we define the distance $d\left(x, y|w\right)$ between categorical data x and y as

$$d\left(x, y|w\right) = \sum_{x_i \neq y_i} w_i \tag{4}$$

LIR: LIR improved I-Relief by using logistic regression, which means iteratively solving w by the expectation maximization (EM) algorithm. The cost function is shown as (5).

$$\min_w \sum_n \log\left(1 + \exp\left(-w^T \bar{z}_n\right)\right) + \lambda \|w\|_1 \tag{5}$$

where

$$\begin{aligned} \bar{z}_n = &\sum_{x_i \in M_n} \left(P\left(x_i = \mathrm{NM}\left(x_n\right)|w\right)\right) d\left(x_n, x_i|w\right) \\ &- \sum_{x_i \in H_n} \left(P\left(x_i = \mathrm{NH}\left(x_n\right)|w\right)\right) d\left(x_n, x_i|w\right) \end{aligned} \tag{6}$$

Here, H_n and M_n are the sets of hit and miss data respectively, and λ is the parameter defined by users.

Relief-based algorithms, especially LIR, can easily adapt to categorical values, as long as there is a definition of the distance between categorical values. Therefore, with the distance function (4), our Relief-based on LIR can deal with categorical values.

Next, a simple example is given to illustrate the multi-dimensional data oriented LIR. Suppose that in the multi-dimensional dataset with three attributes *A*, *B* and *C*, there are three records:

$$x_1 = (A = a1, B = b1, C = c1), \text{negative}$$

$$x_2 = (A = a1, B = b2, C = c2), \text{negative}$$

$$x_3 = (A = a2, B = b1, C = c3), \text{positive}$$

Improved LIR: According to the distance function (4), we can calculate the distribution of $\mathrm{NM}(x_n)$ and $\mathrm{NH}(x_n)$, and then we can calculate $\bar{z}_n$. Take x_1 as an example, we can make $\mathrm{NM}(x_1) = \{x_3\}$ and $\mathrm{NH}(x_1) = \{x_2\}$. For x_1, the NM part of $\bar{z}_1$ is positively correlated with w_A and w_C because the increase of w_A and w_C means that the distance between x_1 and $\mathrm{NM}(x_1)$ increases, so it will be easier to distinguish between the positive case and the negative case. Accordingly, the NH part of $\bar{z}_1$ is negatively correlated with w_B because the decrease of w_B reduces the distance between the two positive cases x_1 and x_2. Besides, $\bar{z}_2$ and $\bar{z}_3$ follow the same correlations. By logistic regression, the correlations are shown in iterations of w (Table IV). Although C is a valuable attribute, it needs to compete with A and B; meanwhile, C has a redundant relationship with A and B respectively. So w_C increases only after w_B has decreased, which shows the role of $\lambda \|w\|_1$ part in (5). Finally, by threshold $\tau_w = 1$, we selected attributes A and C.

TABLE IV
ITERATIONS OF w AFTER EPOCHS

epoch	w_A	w_b	w_c
1	1.0000	1.0000	1.0000
10	1.2037	0.6437	0.8652
100	1.3408	0.2523	0.7730
1000	1.1129	0.0238	1.0013

D. Adapt to Data Imbalance

RC-LIR is superior to LIR in data imbalance cases, where the low frequency of positive examples [19] (here, positive means fault) could significantly impact the algorithm's effectiveness (our challenge 2). Under the influence of an imbalanced training set in the cost function (5), the influence of near hit of positive cases on the weight will become minimal, which makes LIR tend to ignore the commonalities between positive cases, but pay more attention to the near hits between negative cases which are less important in RCA problems. To solve this problem, RC-LIR uses an augmentation coefficient β_n to scale up the faults in the cost function (5) of LIR, as shown in function (7), which enlarges the effect of positive records in training.

E. Adapt to Unseen Root Causes of Faults

For challenge 3 in Section II-C, the key is that if the algorithm totally relies on fault labeling, it cannot discover an attribute that never appears as a fault in the training set. An additional part of RC-LIR will train weights w without fault labeling to improve this. The comparison between (9) and (6) shows our improvement, which is redundant cost part of cost function. The redundant cost part $\alpha \sum_{x_i} \max\left(P\left(x_i = \mathrm{Near}\left(x_n\right)\right) - p_0, 0\right) d\left(x_n, x_i|w\right)$ in RC-LIR can be calculated without data labeling and shows the ability of an attribute to distinguish from surrounding attributes, which means the attribute is unlikely to be redundant. It reveals the value of each attribute when there is no fault, so it has the potential to discover the unseen root causes of faults. The parameter α is used to adjust the weight of this part.

F. The Whole Framework of the RC-LIR

The whole framework of our approach can be summarized as follows.

1) Preprocess and Rule-Based Selecting: Firstly, the sample data is preprocessed. For each piece of data in the sample, to prevent attribute inconsistency, the algorithm will find the complete set of attributes and fill in the default value for the missing data (in our implementation, the default value is an empty string). After that, we do the rule-based selecting in Section III-B

2) RC-LIR Training: Next, we use RC-LIR to train the weight w of each attribute in a logistic regression process. RC-LIR's cost function is shown as (7).

$$\min_w \sum_n \beta_n \log\left(1 + \exp\left(-w^T \bar{z}_n\right)\right) + \lambda \|w\|_1 \tag{7}$$

Among them, β_n and $\bar{z}_n$ is

$$\beta_n = \begin{cases} 1 & \text{if } x_n \text{ is negative} \\ |M_n|/|H_n| & \text{if } x_n \text{ is positive} \end{cases} \tag{8}$$

$$\begin{aligned} \bar{z}_n = & \sum_{x_i \in M_n} \left(P\left(x_i = \mathrm{NM}\left(x_n\right)|w\right)\right) d\left(x_n, x_i|w\right) \\ & - \sum_{x_i \in H_n} \left(P\left(x_i = \mathrm{NH}\left(x_n\right)\right)|w\right) d\left(x_n, x_i|w\right) \\ & + \alpha \sum_{x_i} \max\left(P\left(x_i = \mathrm{Near}\left(x_n\right)\right) - p_0, 0\right) d\left(x_n, x_i|w\right) \end{aligned} \tag{9}$$

Here, $Near\left(x_n\right)$ is the union of $NH\left(x_n\right)$ and $NM\left(x_n\right)$, and the probability threshold p_0 is set to 0.5. The conditional probability $P\left(x_i = \mathrm{NM}\left(x_n\right)|w\right)$, $P\left(x_i = \mathrm{NH}\left(x_n\right)|w\right)$, and $P\left(x_i = \mathrm{Near}\left(x_n\right)\right)$ is estimated through the standard kernel density estimation method in the weighted space.

During the logistic regression process, we initialize w as a zero vector and specify the distance function $d(x, y|w)$ of the categorical value vector as the proportion of unequal values in all values. Our cost function (7) is basically the same as the LIR algorithm (5), except for some differences. We will explain our ideas for making these improvements in the rest of this section. After the logistic regression of w process by the Gradient Descent method, the weight w of every attribute is calculated by RC-LIR. To deal with the dynamic changes in the environment or changes brought about by configuration changes, we set up a regular update mechanism based on the periodicity of the data (i.e., the update period is one day in the scenario).

3) Pass Selected Attributes to Downstream Algorithms: Finally RC-LIR uses a threshold τ_w to control the keeping and discarding of each attribute. Only attributes with w larger than τ_w will be passed to downstream MDRCA algorithms.

G. Summary

According to its principle and our evaluation, we believe that Relief-based algorithms are good for multi-dimensional data attribute selection. However, Relief-based algorithms like LIR also have two aspects of improvement. (i) it does not consider the data imbalance problem, which makes it perform poorly on only a few positive data sets. (ii) it relies heavily on manual labeling, so we can assert that it cannot recognize rare root causes of faults; Our RC-LIR algorithm mainly modified the algorithm for the two points and made two major improvements. Our evaluation in Section IV-C shows the benefits of two improvements to the final effect.

IV. Evaluation

We conducted an extensive experimental study based on real-world data to demonstrate the effectiveness of RC-LIR, aiming to answer the following research questions (RQs):

- RQ1: How effective and efficient is RC-LIR in selecting critical attributes?
- RQ2: How does each component and hyperparameter affect the performance of RC-LIR?
- RQ3: Does the attribute selected by RC-LIR beneficial to downstream MDRCA algorithms?

A. Evaluation Setup

1) Dataset: In this work, we conduct experiments collected from a real-world enterprise. Some basic information is shown in Table V.

TABLE V
Summary of Datasets

System Index	Amount of Logs	Attributes	Faults	$\|\mathcal{R}\|$
1	1083411	45	204	10
2	688025	45	8	6
3	259514	45	196	3
5	844810	45	43	7
6	744467	45	63	8
7	710588	45	19	4
8	393996	45	5	5
9	622984	45	154	9
10	342014	45	5	6
11	653749	45	1	5
12	213418	45	1	5
13	539358	45	227	3
15	640059	45	74	10

For the problem of RCOAS, the data we collected has the following advantages:

1) For the attribute selection problem, the dataset has up to 45 attributes and has strong data diversity, which can cover all the situations mentioned in Section II;
2) The data are collected from 15 different systems. Data from both 15 systems have the same 45 attributes, while each system has its own format and characteristic of values, so the selection result should be different in different systems. The same 45 attributes can also prevent the attribute selection algorithms from simply classifying the data into systems and then answering;
3) Data from every system have completed 45 attributes but in different forms. About 2.8% of the points can be identified as failure or timeout, enabling us to analyze different systems' characteristics and mark the fault manually;
4) Root causes and critical attributes of every system have been manually labeled.

TABLE VI
F1-SCORE COMPARISON OF ATTRIBUTE SELECTION ALGORITHMSS

Algorithm	F1-score	Recall	Precision	Elapsed Time (s)
Using All Attributes	0.2500	**1.0000**	0.1429	-
Laplacian Score	0.5739	0.7332	0.4714	4.3
Top10 KL Divergence	0.2065	0.2638	0.1696	**1.2**
Top5 KL Divergence	0.2751	0.2445	0.3144	**1.2**
Best-of-Adtributor	0.6895	0.5373	0.9621	6.6
Boruta	0.5419	0.6197	0.4814	16.5
Relief	0.5691	0.6706	0.4942	148.0
Logistic I-Relief	0.7240	0.6424	0.8293	28.6
RC-LIR (τ_w=0.2)	0.8692	0.8413	0.8990	25.3
RC-LIR (τ_w=0.5)	0.8642	0.8248	0.9075	25.6
RC-LIR (τ_w=1.0)	**0.8788**	0.8115	0.9583	24.6
RC-LIR (τ_w=2.0)	0.8582	0.7703	0.9687	23.8
RC-LIR (τ_w=5.0)	0.8457	0.7471	**0.9742**	25.0

As stated in Section II-B, we evaluate the performance of RCOAS at the fault case level. There are 1000 fault cases in total. The root cause attributes and values of these fault cases are carefully labeled to accurately evaluate the performance of RC-LIR, baseline methods, and downstream algorithms. Four experienced operators completed the root cause labeling process: Two operators find the anomalies and label the root causes independently; When their labels diverge, the third operator is involved and judges independently. The fourth operator supervises and checks the whole process and makes decisions when the other three take different opinions. The entire labeling process took nearly a month.

2) Baselines: Because we are considering the RCA process of multi-dimensional data, the existing FS baseline may not be completely suitable, and the selection of baseline is hard to deal with.

For filter methods, we choose two widely used algorithms, KL Divergence and Laplacian Score. For the KL Divergence, it is difficult to determine a unified threshold for each system, so we individually choose the top5 or the top10 as two baselines, in reference to a common solution. For the Laplacian Score, we adjust its threshold to maximize its overall F1-score.

As for embedded methods, common ones are SVM or Random Forest, which can tell feature importance, but these algorithms may not be suitable for multi-dimensional data. Instead, we hope to find an RCA algorithm that can tell the importance of features, so we can select attributes according to their scores with a threshold. Note that most RCA algorithms encounter great difficulties in dealing with our 45-dimensional dataset $\mathscr{D}$ due to the aforementioned curse of dimensionality, so we have little choice. Finally, we choose the Adtributor [1] algorithm, which runs impressively fast, and then construct the Best-of-Adtributor attribute selection based on the scoring of attributes as an embedded method for multi-dimensional data.

Among wrapper methods, we choose the Boruta algorithm, which has been widely used and considered effective. Another important reason why we choose Boruta is that, among all kinds of wrapper methods, Boruta is one of the few algorithms that support the categorical values. In terms of implementation details, we basically adopt BorutaPy implementation and params, only adjusting some code to make it compatible with categorical values of string type.

Of course, to show that our improvement on LIR is effective, we also add Relief and LIR as a baseline. In addition, we also consider using all attributes without any filter as evidence of the effectiveness of our RC-LIR algorithm.

The settings of our baselines are tuned to reach as higher F1-scores in the dataset as possible:

1) KL-Divergence: pick the top 5 or 10
2) Boruta: $n_{\text{estimators}} = 1000$, perc $= 75$, $\alpha = 0.05$, $\text{max}_{\text{iter}} = 200$
3) Relief: $n_{\text{epoch}} = 200$, $\tau = 0.3$
4) LIR: $n_{\text{epoch}} = 2000$, lr $= 0.01$, $\tau = 2.5$, $\lambda = 0.01$
5) Others: default or no parameters

3) Evaluation Metric: We use F_1-$Score$ to evaluate the performance RCOAS approaches and the downstream MDRCA algorithms. It is calculated based on attributes, so if an attribute is selected or specified by the algorithm and it is in the standard results, it will be a true positive (TP); if the attribute is not in the standard results, it is a false positive (FP); if the algorithm misses one attribute in the standard results, it is a false negative (FN). Then, among all the test cases, F_1-$Score$ is calculated as (10):

$$\begin{aligned} precision &= \frac{\#TP}{\#TP + \#FP} \\ recall &= \frac{\#TP}{\#TP + \#FN} \\ F_1\text{-}Score &= \frac{2 * precision * recall}{precision + recall} \end{aligned} \tag{10}$$

For RQ1 & RQ2, we use $precision$, $recall$, and F_1-$Score$. The target is critical attributes, which is the union of all fault cases of the system.

For RQ3, we use F_1-$Score$. The target attributes are directly the root cause attributes labeled by operators.

4) Implementation: During all the evaluations, we run every experiment on 12×Intel(R) Core(R) i9-10920X@3.5GHz CPU, 64GiB RAM, and Geforce RTX 3080 GPU. All algorithms are implemented with Python.

B. Evaluation of Critical Attribute Prediction (RQ1)

For the first experiment, we directly compared the results of attribute selection with the critical attributes of manual labeling to get the recall, precision, and F1-score. In this experiment, recall and precision have different practical meanings: a higher recall rate means that the process of attribute selection will not cause valuable attributes to be ignored, while a higher precision rate means fewer redundant attributes so that the efficiency and performance of downstream algorithms can be improved.

The experimental results are shown in Table VI. We can see that under the appropriate threshold, our RC-LIR algorithm can achieve the highest F1-score, which proves that the algorithm is effective. At the same time, with some adjustments of

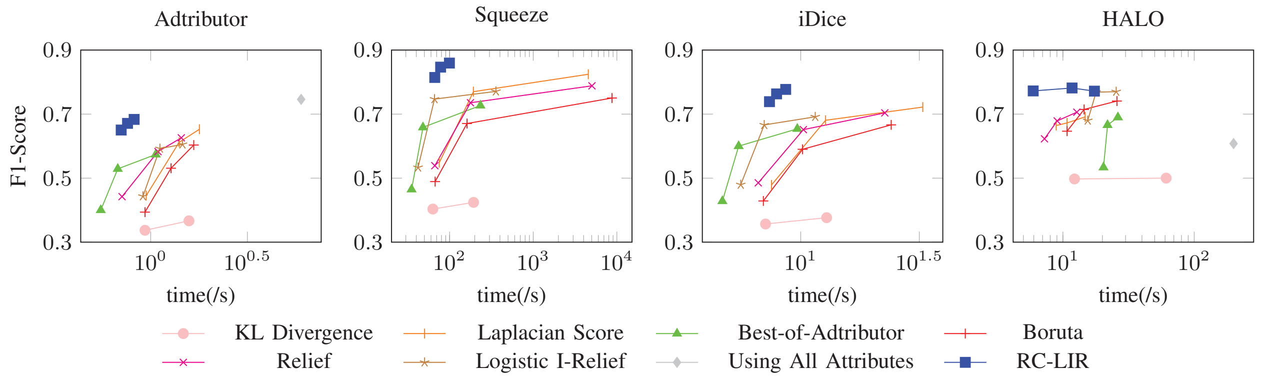

Fig. 5. Performance of RCA algorithms with different attribute selection algorithms.

the threshold, our RC-LIR algorithm can achieve great recall or precision under the condition of ensuring the overall effect, which can adapt to the needs of different scenes. In terms of time consumption, there is no significant difference between the attribute selection algorithms, and the attribute selection process only needs to be carried out once when the whole multi-dimensional data analysis system is started. Therefore, we think that the time consumption of RC-LIR is satisfactory.

C. Ablation Study and Hyperparameter Sensitivity (RQ2)

Components. We will show that our two significant improvements to LIR are effective. Therefore, we design three kinds of baseline algorithms based on RC-LIR: the first does not contain improvement Section III-D (which means β_n is always 1 but not calculated by (8)), and the second does not contain improvement III-E so that they are concerned only about supervised learning (which means α in (9) is 0), and the last ignores supervised learning (which means α in (9) is a large number). Among all the tests, we adjust other parameters like learning rate to get the best F1-score. The experimental results are shown in Table VII, which shows that our two main improvements to LIR are effective.

Hyperparameters. There are two parameters that need to be pre-configured for RC-LIR: τ_w as the threshold to accept w and α in (9). τ_w decides how likely to accept an attribute. Lower τ_w means higher recall and lower precision, and vice versa. The corresponding experimental results are shown in Table VI. Here, as our determination is aimed to get an F1-score as higher as possible, we set τ_w as 1. α decides the weights of the supervised learning and the unsupervised learning parts in (9). Through experiment results in Table VII, we set α as 0.5, and we also found that α between 0.1 and 10 has little impact on the results.

D. Evaluation of Downstream MDRCA Algorithms (RQ3)

We carefully choose four MDRCA algorithms for the sake of their effectiveness and efficiency, i.e., Squeeze [4] as a state-of-the-art MDRCA algorithm, Adtributor [1] with short elapsed time, and HALO [8], iDice [2] with both aspects. Because the Squeeze and iDice are too slow or even crash when facing data with all the 45 attributes, we only evaluate *Using All Attributes* on Adtributor and HALO.

The experimental results are shown in Fig. 5. In each subfigure, every line represents a combination of RCOAS and RCA. The X-axis (logarithmic) is the average (for all 1,000 cases) time overhead of RCA, and Y-axis is the average F1-score of RCA. Every combination has up to 3 measurement points for different settings of each RCOAS method, which can provide a detailed comparison. (e.g., estimate the F1-scores for the same time overhead). Take HALO as an example, Compared to Using All Attributes, RC-LIR reduces the time overhead from 199.6s to 5.9s and reaches even higher F1.

It can be seen that our RC-LIR algorithm can achieve a better F1-score under the same elapsed time, and it can greatly shorten the time consumption of downstream algorithms under the same effect of the downstream algorithm. We can also see that the more obvious the curse of dimensionality of the downstream RCA algorithm, the greater the advantage of our RC-LIR algorithm compared with other FS algorithms.

TABLE VII
F1-SCORE COMPARISON OF DIFFERENT PARAMS

α	β	F1-score	Recall	Precision
0.0	calculated by (8)	0.7667	0.6805	0.8780
0.01	calculated by (8)	0.7844	0.7355	0.8404
0.1	calculated by (8)	0.8534	0.8150	0.8957
0.3	calculated by (8)	0.8696	0.8158	0.9310
0.5	calculated by (8)	**0.8788**	0.8115	**0.9583**
1.0	calculated by (8)	0.7840	**0.8629**	0.7184
10.0	calculated by (8)	0.6640	0.8334	0.5519
100.0	calculated by (8)	0.6068	0.7575	0.5061
0.1	always 1	0.7097	0.6357	0.8031
0.3	always 1	0.7260	0.6343	0.8488
0.5	always 1	0.7530	0.6754	0.8507
1.0	always 1	0.6899	0.6747	0.7058

V. Discussion and Future Work

From evaluation, we prove that RC-LIR is effective on RCOAS. Compared with the existing FS algorithms, RC-LIR can select the critical attributes more accurately, and the process of attribute selection does have a favorable impact on the downstream multi-dimensional data analysis algorithm.

TABLE VIII
ADVANTAGES AND LIMITATIONS OF EXISTING SOLUTIONS

Algorithms	Generalization	Categorical Values	Support Wide RCA Algorithms	Suitable for Incomplete Training Set	Method
Information Divergence [16]	**Yes**	**Support**	**Yes**	No	Filter
Laplacian Score [9]	**Yes**	Not Support	**Yes**	**Yes**	Filter
Relief [7]	**Yes**	**Support**	**Yes**	No	Filter
Boruta [14]	**Yes**	**Support**	No	No	Wrapper
Embedded Methods	**Yes**	Depends on Implementation	No	No	Embedded
RC-LIR	**Yes**	**Support**	**Yes**	**Yes**	Filter

Some evaluation results also attracted our attention. The Best-of-Adtributor algorithm we constructed also gets quite high precision since most of the algorithm's results are the root causes from the small training set, so it can only learn from the alerts that have occurred in a short period of time. Most root causes from these alerts are surely the critical attributes, but the root causes that are not in this period will be ignored. Therefore, the Best-of-Adtributor algorithm has high precision and low recall, which is in line with our expectations. However, in the experiment, we also found that if the training set is large enough, this kind of attribute selection which directly calls the RCA algorithm can also have a good effect. In the future, we may be able to construct RCA algorithms that are efficient for high-dimensional datasets to select attributes.

We also notice that different downstream algorithms have different requirements for attribute selection. For example, when the downstream algorithm is iDice, which is fast to a certain extent, we can accept that the result of attribute selection has as many as 20 attributes. But for Squeeze, the efficiency of 20 attributes in practical application is likely to be unacceptable. This inspires us to increase the customization of attribute selection results. A simple solution can be to output the weight of each attribute so that downstream algorithms can make choices according to their own needs.

In addition, in our experimental setup, feature engineering can only carry out feature selection but not feature transformation and merging. However, we can expect that the more complex feature engineering applied to multi-dimensional data can better meet the needs of the downstream algorithms, which could be a huge improvement.

VI. Related Works

This section introduces some common feature selection methods in the ML field. Based on the interaction with the learning model, feature selection methods can be mainly grouped into three types: filter, wrapper, and embedded methods. We will introduce their general principles and shortcomings.

A. Filter Methods

Some common feature selection algorithms are based on a filter, which means that they first calculate the weight of each feature according to its distribution and then select the retained features according to the weight. Some typical algorithms are based on Information Divergence [16]. The algorithm calculates the information gain of each feature to the classification result as the weight of the feature. There is also a better algorithm based on Fisher Score [20] or Laplacian Score [9], which calculates the score of features according to the following two principles: (i) the variance of features should be as large as possible; (ii) features should reflect continuity within the same category.

Filter methods may have the following problems on RCOAS. First, independently calculating the weight of each attribute means that the algorithm cannot consider multiple attributes comprehensively, which may be reasonable in the ML field, but for RCA algorithms, it is likely that multiple effective features with mutual redundancy will be regarded as good attributes. Secondly, a large number of filter methods based on statistics rely on the classification results for weight calculation, which means that their selected features can only reflect the situation of training data, which is very reasonable in the ML field, but for RCA algorithms, it means that the whole analysis process cannot deal with unseen patterns of faults. [21] Finally, like the Laplacian Score [22] algorithm, many algorithms cannot adapt to the categorical values in the data well, and the categorical values in the multi-dimensional data are very common and of great significance.

There are also many filter methods based on the Monte Carlo algorithm [23]. One approach is to turn the problem into a high-dimensional problem and use the simulated annealing process to select features. The other approach is to take feature selection as a decision and use reinforcement learning to select features. The common feature of these Monte Carlo algorithms is that they all need a specific downstream algorithm to evaluate the solution (in this case, the solution is a set of features), which means that the feature selection is customized for this downstream algorithm, and it may not work well for other downstream algorithms. In addition, due to the high time cost of the RCA algorithm, repeated calls to downstream algorithms may also bring high time complexity to the feature selection process.

Relief [7] is a classical method of feature selection. It is also a Monte Carlo method, but several variants of Relief [17], [24] are not Monte Carlo methods. According to our evaluation in Section IV, Relief-based algorithms, especially LIR, are suitable for the multi-dimensional data attribute selection problem to a certain extent. However, LIR still has many characteristics that are unsuitable for multi-dimensional data attribute selection, which has been introduced in Section III.

B. Embedded Methods

Most embedded methods construct a specific classifier to train the classifier and realize feature selection at the same time. The most typical embedded idea is to use some algorithms that can naturally output feature weights, such as SVM [10], Random Forest [25], XGBoost [11], [12], etc., and directly use the weight output by the algorithm to complete feature selection. However, when directly applied to multi-dimensional data attribute selection, our experiments show that the important evaluation of RCA algorithms is not consistent with the classifier inside the embedded methods, which makes the effect of attribute selection unsatisfactory. In addition, the application of SVM, Random Forest, XGBoost, and other algorithms to the RCA process is also prone to problems such as the inability to deal with unseen patterns of faults, difficulty dealing with categorical data, and so on.

C. Wrapper Methods

Wrapper methods select the best feature subset based on a specific classifier's results. The Boruta [13], [14] algorithm is a widely used wrapper FS method, which means that in the process of the algorithm, it will repeatedly call downstream algorithms to process feature selection. The Boruta algorithm is very effective, but it has two serious defects for the downstream MDRCA process. First, Boruta needs to call a downstream algorithm to score the importance of features. In real-world processing, it may be impossibly difficult to find a suitable algorithm. Even if it is found, it may not be suitable for a variety of data. Second, because the downstream algorithms for outputting the importance of features here are usually Random Forests, XGBoost, etc., Boruta also has similar problems as the embedded methods mentioned above. That is, it cannot deal with unseen patterns of faults.

VII. Conclusion

Multi-dimensional root cause analysis is of critical importance for many online software services, but the performance of existing algorithms is severely affected by the curse of dimensionality. In this work, we formally introduce the task of RCOAS to ease the problem of too many dimensions for a variety of MDRCA algorithms. Moreover, we conduct an empirical study to reveal the insufficient of native feature selection methods. Inspired by the study, we propose RC-LIR, an RCOAS approach. Our main application area is time-sensitive operation scenarios, including banks, cloud services, and large-scale data centers. In our time-sensitive operation scenarios, hundreds of seconds of latency are significant. RC-LIR combines rule-based selecting and a improved feature selection approach. The evaluation based on real-world data results shows that RC-LIR can achieve an F1-score of 0.8788, outperforming all the baseline approaches. Furthermore, our experiments show that RC-LIR can improve the effectiveness and efficiency of MDRCA algorithms.

Acknowledgment

This work is supported by the National Key R&D Program of China 2019YFB1802504, and the State Key Program of National Natural Science of China under Grant 62072264, and the Natural Science Foundation of Tianjin under Grant No. 21JCQNJC00180.

References

[1] R. Bhagwan, R. Kumar, R. Ramjee, G. Varghese, S. Mohapatra, H. Manoharan, and P. Shah, "Adtributor: Revenue debugging in advertising systems," in *11th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 14)*, 2014, pp. 43–55.

[2] Q. Lin, J.-G. Lou, H. Zhang, and D. Zhang, "idice: problem identification for emerging issues," in *Proceedings of the 38th International Conference on Software Engineering*, 2016, pp. 214–224.

[3] Y. Sun, Y. Zhao, Y. Su, D. Liu, X. Nie, Y. Meng, S. Cheng, D. Pei, S. Zhang, X. Qu *et al.*, "Hotspot: Anomaly localization for additive kpis with multi-dimensional attributes," *IEEE Access*, vol. 6, pp. 10 909–10 923, 2018.

[4] Z. Li, C. Luo, Y. Zhao, Y. Sun, K. Sui, X. Wang, D. Liu, X. Jin, Q. Wang, and D. Pei, "Generic and robust localization of multi-dimensional root causes," in *2019 IEEE 30th International Symposium on Software Reliability Engineering (ISSRE)*. IEEE, 2019, pp. 47–57.

[5] C. Borgelt and R. Kruse, "Induction of association rules: Apriori implementation," in *Compstat*. Springer, 2002, pp. 395–400.

[6] M. Al-Maolegi and B. Arkok, "An improved apriori algorithm for association rules," *arXiv preprint arXiv:1403.3948*, 2014.

[7] I. Kononenko, "Estimating attributes: Analysis and extensions of relief," in *European conference on machine learning*. Springer, 1994, pp. 171–182.

[8] X. Zhang, C. Du, Y. Li, Y. Xu, H. Zhang, S. Qin, Z. Li, Q. Lin, Y. Dang, A. Zhou *et al.*, "Halo: Hierarchy-aware fault localization for cloud systems," in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 2021, pp. 3948–3958.

[9] X. He, D. Cai, and P. Niyogi, "Laplacian score for feature selection," *Advances in neural information processing systems*, vol. 18, 2005.

[10] M. Pal and G. M. Foody, "Feature selection for classification of hyperspectral data by svm," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 48, no. 5, pp. 2297–2307, 2010.

[11] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd acm sigkdd international conference on knowledge discover and data mining*, 2016, pp. 785–794.

[12] Y. Wang and X. S. Ni, "A xgboost risk model via feature selection and bayesian hyper-parameter optimization," *arXiv preprint arXiv:1901.08433*, 2019.

[13] M. B. Kursa, A. Jankowski, and W. R. Rudnicki, "Boruta–a system for feature selection," *Fundamenta Informaticae*, vol. 101, no. 4, pp. 271–285, 2010.

[14] M. B. Kursa, W. R. Rudnicki *et al.*, "Feature selection with the boruta package," *J Stat Softw*, vol. 36, no. 11, pp. 1–13, 2010.

[15] L. Yin, Y. Ge, K. Xiao, X. Wang, and X. Quan, "Feature selection for high-dimensional imbalanced data," *Neurocomputing*, vol. 105, pp. 3–11, 2013.

[16] C. Lee and G. G. Lee, "Information gain and divergence-based feature selection for machine learning-based text categorization," *Information processing & management*, vol. 42, no. 1, pp. 155–165, 2006.

[17] Y. Sun, "Iterative relief for feature weighting: algorithms, theories, and applications," *IEEE transactions on pattern analysis and machine intelligence*, vol. 29, no. 6, pp. 1035–1051, 2007.

[18] Y. Sun and J. Li, "Iterative relief for feature weighting," in *Proceedings of the 23rd international conference on Machine learning*, 2006, pp. 913–920.

[19] C. C. Teck, L. Xiang, Z. Junhong, L. Xiaoli, C. Hong, and D. Woon, "Hybrid rebalancing approach to handle imbalanced dataset for fault diagnosis in manufacturing systems," in *2012 7th IEEE Conference on Industrial Electronics and Applications (ICIEA)*. IEEE, 2012, pp. 1224–1229.

[20] Q. Gu, Z. Li, and J. Han, "Generalized fisher score for feature selection," *arXiv preprint arXiv:1202.3725*, 2012.

[21] J. Song, H. Takakura, Y. Okabe, and K. Nakao, "Toward a more practical unsupervised anomaly detection system," *Information Sciences*, vol. 231, pp. 4–14, 2013.

[22] R. Huang, W. Jiang, and G. Sun, "Manifold-based constraint laplacian score for multi-label feature selection," *Pattern Recognition Letters*, vol. 112, pp. 346–352, 2018.

[23] R. Gaudel and M. Sebag, "Feature selection as a one-player game," in *International Conference on Machine Learning*, 2010, pp. 359–366.

[24] B. Tang and L. Zhang, "Local preserving logistic i-relief for semi-supervised feature selection," *Neurocomputing*, vol. 399, pp. 48–64, 2020.

[25] M. A. M. Hasan, M. Nasser, S. Ahmad, and K. I. Molla, "Feature selection for intrusion detection using random forest," *Journal of information security*, vol. 7, no. 3, pp. 129–140, 2016.