Adaptive Algorithms for Diagnosing Large-Scale Failures in Computer Networks

Srikar Tati *, Bong Jun Ko†, Guohong Cao*, Ananthram Swami‡, and Thomas La Porta *
*Institute for Networking and Security Research, Pennsylvania State University, Email: {tati,gcao,tlp}@cse.psu.edu
†IBM TJ Watson, Email: bongjun_ko@us.ibm.com;‡Army Research Laboratory, Email: ananthram.swami.civ@mail.mil

Abstract—We propose a greedy algorithm, Cluster-MAX-COVERAGE (CMC), to efficiently diagnose large-scale clustered failures. We primarily address the challenge of determining faults with incomplete symptoms. CMC makes novel use of both positive and negative symptoms to output a hypothesis list with a low number of false negatives and false positives quickly. CMC requires reports from about half as many nodes as other existing algorithms to determine failures with 100% accuracy. Moreover, CMC accomplishes this gain significantly faster (sometimes by two orders of magnitude) than an algorithm that matches its accuracy. When there are fewer positive and negative symptoms at a reporting node, CMC performs much better than existing algorithms. We also propose an adaptive algorithm called Adaptive-MAX-COVERAGE (AMC) that performs efficiently during both independent and clustered failures. During a series of failures that include both independent and clustered, AMC results in a reduced number of false negatives and false positives.

Keywords—Fault diagnosis; Large-scale failures; Incomplete information; Clustered failures

1 INTRODUCTION

Critical steps in achieving high network reliability are detecting, diagnosing, and localizing faulty network elements [1] (e.g., network nodes, links) after failures. Conventional fault detection and diagnosis techniques [2], [3] deal with failures that are sporadic and independent in nature, i.e., caused by individual component failures (e.g., broken links, adapter failure, routing software malfunction, etc.). In this paper, we focus on large-scale failures in computer networks that span different ISPs and collaborative networks with different access policies.

Large-scale failures are possible due to events such as intentional attacks and natural disasters. During natural disasters like Hurricanes Katrina and Irene [4], earthquakes in Japan and Taiwan [5], and electricity blackouts [4], there are reports that large portions of the Internet are damaged. In addition, intentional attacks (e.g., 9/11 attack [6]) may damage a significant portion of a network, and create massive disruptions. We note that the types of outages caused by large-scale failures differ greatly from those caused by typical equipment faults [2], [3], [7]. Large-scale outages tend to create faults at multiple components that are geographically close to each other. We call these failures clustered failures.

There is considerable interest in large-scale failures in the current literature [8], [9], but not much work focuses on diagnosing these clustered failures. Due to discrepancy in failure patterns, the performance of fault diagnosis techniques that are focused on independent failures [3], [7] degrades when applied to clustered failures. A fault diagnosis algorithm called netCSI [10] is proposed to localize large-scale failures. It is shown that by considering the failure patterns of large-scale outages, this algorithm can achieve higher accuracy than existing algorithms developed for independent failures [7]. However, the drawback of netCSI is that the runtime complexity of the algorithm increases exponentially with increase in network size since it is a combinatorial approach.

In this paper, we propose a new algorithm called Cluster-MAX-COVERAGE (CMC) that diagnoses large-scale clustered failures1. To identify the faulty network elements (i.e., network nodes, routers, and links) CMC utilizes a knowledge base of possible network paths and end-to-end symptom information. The observed end-to-end symptoms during failures include both negative symptoms, such as which source-destination pairs are disconnected, as well as positive symptoms, such as which source-destination pairs can still communicate. This information is reported to the network manager by a few selected nodes in the network called reporting nodes; a complete list of symptoms is not required. Using this information, CMC outputs a hypothesis list which consists of a set of network elements whose failures are consistent with the symptoms.

To solve the issue of run-time complexity, CMC adopts a greedy approach when generating the hypothesis list of faulty network elements, as opposed to the combinatorial approach in netCSI. Our greedy approach is similar to a fault diagnosis algorithm called MAX-COVERAGE (MC) [7], which is developed to diagnose black holes or silent failures (independent failures) in IP networks. During clustered failures, the performance of

1. This paper is extension of a conference version [11].
MC degrades significantly—in particular it produces a prohibitively high number of false negatives (see Section 6.3.1). To overcome this limitation, CMC uses clusters of objects instead of single objects when generating the hypothesis list.

The major contributions of CMC include:

- **Utilizing positive symptoms**: Unlike MC which uses only negative symptoms, CMC considers both positive and negative symptoms. As a result, CMC outputs a hypothesis list with fewer false positives. In our simulations that are carried out on both synthetic and realistic network topology datasets, false positives are reduced on average by 45% with inclusion of positive symptoms.

- **Coping with partial symptoms due to varying number of reporting nodes**: Unlike the existing algorithms [2], [3], [7] that assume availability of complete information, CMC specifically addresses the issue of incomplete symptoms due to a limited reporting nodes. During large-scale failures, it is unlikely that complete symptoms are available because some of the reporting nodes may not be able to reach the network manager due to either failures, heterogenous ownership or resource and time constraints. In our simulation results, when compared to MC, only 10% of the reporting nodes are required by CMC in a realistic topology to achieve 100% accuracy.

- **Coping with partial symptoms at a reporting node**: In addition, we also explore the case in which reporting nodes themselves may only have partial symptoms. We find that CMC is more accurate than MC and netCSI, when there are fewer negative and positive symptoms at a chosen reporting node. We observe this in several scenarios in our simulations—in particular, with 30% of partial symptoms at 40 reporting nodes, CMC achieves almost 100% accuracy, whereas netCSI and MC provide 70% and 30% accuracy, respectively.

- **Ranking scheme**: We develop a ranking scheme as part of CMC to post-process the hypothesis list. With ranking, CMC decreases the false positives in the hypothesis list.

To show the benefits of our algorithm, we compare CMC with two state-of-the-art algorithms (MC and netCSI) through various simulations (see Section 6.3). We observe that under clustered failures, CMC outputs a hypothesis list with a significantly lower number of false negatives and a similar or slightly higher number of false positives than MC. In most failure scenarios, we believe that false negatives are far more significant than false positives, because they may go completely undetected or may be very expensive to localize. Therefore, to decrease the number of false negatives during large-scale failures, CMC is willing to pay a small extra price in terms of false positives. Also, when compared to netCSI, the run-time complexity of CMC is dramatically reduced while achieving the same level of accuracy—for example, the run-time of CMC is lower than the run-time of netCSI by up to two orders of magnitude when the number of reporting nodes is small (see Section 1.6 of the supplemental file [12]).

While CMC is better suited for diagnosing large-scale clustered failures, we also observe that MC performs better in localizing independent failures. Therefore, we also propose Adaptive-MAX-COVERAGE (AMC), which employs a Bayesian decision technique as a pre-processing step to classify network failures into independent or clustered based on end-to-end symptoms. AMC employs either MC or CMC depending on the type of failure. Our simulation results show the effectiveness of this adaptive approach over a series of both independent and clustered failures.

This is the main file which covers the key ideas and concepts of our solution and presents important results of our evaluation. The other auxiliary information like pseudo-code of the algorithm, supporting results, discussion of results etc in the supplemental file [12].

The remainder of the main file is organized as follows. We describe existing algorithms, relevant to our work, in Section 3. We present a system model and an example to illustrate the high-level concepts and novel aspects of our algorithm in Section 4. Then, the three phases of CMC are described in Section 5, and the evaluation of CMC through simulation study is given in 6. We propose the adaptive algorithm AMC in Section 7, and provide results of a simulation study of AMC in Section 8. Finally, we describe related work in Section 2 and conclude the paper in Section 9.

2 RELATED WORK

There is much prior work [1], [2], [7], [13] that address the localization of independent failures, but there is little focus on diagnosing large-scale clustered failures. However, there is considerable interest on other aspects of large-scale failures in the current literature [8], [9].

Recently, netCSI [10], was proposed to diagnose large-scale failures. It generates possible causes of a failure. However, there is a limitation of run-time in large networks due to the combinatorial nature of the algorithm. We propose Cluster MAX-COVERAGE (CMC), to diagnose massive failures accurately and solve the issue of run-time.

MAX-COVERAGE (MC) [7], a fault diagnosis algorithm based on SCORE (space correlation engine) [2], focuses on MPLS-over-IP backbone networks. Localization agents use end-to-end connectivity measurements as alarms and employ spatial correlation techniques to isolate the failures. Since these algorithms diagnose independent failures, their performance degrades for large-scale failures when there are incomplete symptoms. CMC addresses the problem of incomplete symptoms under large-scale failures using both negative and posi-

2. In our simulations, we select reporting nodes randomly.
tive symptoms, unlike MC and SCORE, which consider only negative symptoms.

Sherlock [13], Shrink [3], Steinder et al. [14], and netCSI [10] propose fault diagnosis techniques that use combinatorial algorithms with exponential computational complexity. They propose various optimizations to minimize the complexity of the algorithm. However, in large networks the complexity becomes very high. CMC which is based on a greedy approach has lower computational complexity than combinatorial approaches. Furthermore, both Sherlock and Shrink assume the availability of complete symptoms, whereas CMC performs well even with incomplete symptoms.

Moreover, in this paper, we recognize the advantages of having different fault diagnosis algorithms under different failures, and propose Adaptive MAX-COVERAGE (AMC) algorithm. To the best of our knowledge, AMC is the first algorithm to focus on diagnosing both independent and clustered failures when there are partial symptoms. In addition, we provide an extended literature review in Section 3 of the supplemental file [12] comparing our work with network tomography based algorithms.

In the next section, we provide a background for two existing fault diagnosis algorithms, MAX-COVERAGE (MC) and netCSI that are closely related to our algorithm, along with their limitations under large-scale failures.

3 BACKGROUND
MAX-COVERAGE (MC) [2], [7] addresses a problem similar to ours, but is proposed to diagnose black holes or silent failures which are independent. MC is based on a greedy algorithm that considers only a set of negative symptoms called a failure signature. It uses a metric called link coverage (similar to BC, betweenness centrality, in Section 4.2), which is defined as the number of negative symptoms that are explained due to the failure of a given object. This metric is defined for all objects and used as a heuristic to select possible causes of a failure. MC can be described briefly in three steps. The first step is to iteratively pick an object that has the maximum link coverage. The second step is to remove the symptoms from the failure signature that are explained by this object (picked in the first step). The third step is to repeat this process until there are no unexplained negative symptoms in the failure signature. Finally, MC outputs a minimum set of objects that explains all given negative symptoms in the failure signature.

MC localizes both single and multiple independent failures. It diagnoses multiple simultaneous failures by dividing the set of a failure signature into different subsets, and each set is explained by one of the multiple failures. However, in large-scale clustered failures, there is a large number of simultaneous failures which are geographically close to each other. In addition, there is high overlap among negative symptoms that are explained by failure objects. Hence, under clustered failures, the explanation with a minimum set of objects will miss failures, and thus MC may result in a large number of false negatives. As explained in Section 1, a high number of false negatives is unacceptable during large-scale failures, since the network manager cannot localize them all or must incur a high cost to diagnose them.

netCSI [10], which is based on a combinatorial approach, was proposed to diagnose large-scale failures. This is a two step algorithm that generates a hypotheses list which has different sets of objects that explain the symptoms caused by the failures. In addition, a ranking algorithm is presented which produces a ranked list of the sets of objects. netCSI yields large gains in accuracy over MC when there is a limited number of reporting nodes, but this comes at the cost of increased run-time. The high run-time is due to the exponential complexity of the combinatorial algorithm. This is undesirable, especially in the case of large networks.

4 PROBLEM AND MOTIVATION
In this section, we present our system model and the network measures used in our algorithm, and describe our approach at a high level through an example. Our algorithm diagnoses large-scale failures using the following inputs: the path information (topology) and end-to-end symptoms from reporting nodes.

4.1 System Model
We consider a network with \( n \) nodes, represented by the set \( N \), and \( l \) links, represented by the set \( L \). We define the term objects to represent either nodes or links. The set of objects is denoted by \( O \) which is the union of \( N \) and \( L \). In addition, these objects are divided into clusters based on a specific attribute that is known to the network manager. There are \( c \) clusters in total, each of which can have a variable number of objects, and are represented by \( C = \{ C_1, C_2, C_3, \ldots, C_c \} \). Moreover, these clusters are not mutually exclusive and can overlap with each other.

We assume nodes communicate with each other over multiple routes. The set of source nodes is represented by \( S = \{ s_1, s_2, s_3, \ldots, s_m \} \), and \( S \subseteq N \). The set of \( k \) destination nodes for a source \( s_i \in S \) is denoted by \( D_i = \{ d_1, d_2, \ldots, d_k \} \). The routes from source node \( s_i \) to a destination node \( d_j \in D_i \) are given by \( X_{i,j} = \{ x_1^{s_i,j}, x_2^{s_i,j}, \ldots, x_r^{s_i,j} \} \). The number of paths \( r \) between a given source and destination pair is not assumed to be constant; it depends on \( s_i \) and \( d_j \), but we suppress the notation for clarity. The \( q \)-th route is represented by \( x_q^{s_i,j} \) and is stored as a set of objects in \( O \) that are present along the route. If any of the objects along a route is faulty, then that route is disconnected; otherwise it is considered a connected route. This path information between a source node \( s_i \in S \) and its corresponding destination nodes is collected periodically, and updates are sent to the network manager.

The end-to-end symptom information represents sources’ connectivity and dis-connectivity to their respective destination nodes. A destination node \( d_j \in D_i \)
is regarded as connected to $s_i$ if there exists at least one connected route in $X_{i,j}$, and disconnected if all the routes in $X_{i,j}$ have at least one failure. Symptoms corresponding to a source node $s_i \in S$ can be collected by probing destination nodes that are present in $D_i$.

Source nodes that report end-to-end symptoms and the path information to the network manager are called reporting nodes. In practice, the collection of symptom information involves high overhead in terms of network management traffic. In addition, gathering information from all reporting nodes during a massive failure may not be possible. Hence we address the fault diagnosis problem under partial information due to limited reporting nodes in the network, not assuming the availability of the complete end-to-end symptoms from all nodes.

### 4.2 Network Measures

We use two important network measures from graph theory: betweenness centrality (BC) and group betweenness centrality (GBC). BC for an object $o_i \in O$ is defined as the fraction of paths that pass through $o_i$ out of all the paths between different nodes in the network. In the above definition, the considered paths between nodes are those which are available to the network manager from reporting nodes. Moreover, the paths are assumed to be loop-free. GBC is a network measure for a cluster of objects, $C_i \in C$, in the network. It is defined as the fraction of the paths that pass through $C_i$ out of all the paths between different nodes in the network. The mathematical definitions for BC ($o_i$) and GBC ($C_i$) are,

$$BC(o_i) = \sum_{n_j \neq n_k} \frac{p_{n_j,n_k}(o_i)}{p_{n_j,n_k}}$$

$$GBC(C_i) = \sum_{n_j \neq n_k} \frac{p_{n_j,n_k}(C_i)}{p_{n_j,n_k}}$$

where $p_{n_j,n_k}$ is the total number of the paths between nodes $n_j$ and $n_k$, $p_{n_j,n_k}(o_i)$ is the number of paths between nodes $n_j$ and $n_k$ that pass through $o_i$, and $p_{n_j,n_k}(C_i)$ is the number of paths between nodes $n_j$ and $n_k$ that pass through at least one object in $C_i$. BC is used in AMC while analyzing the failure data (Section 7.2), and GBC is used in TATI (Section 5.2), and GBC is used in AMC while analyzing the failure data (Section 7.2).

### 4.3 Illustrative Example

Consider the network in Figure 1, in which there are two sources, $S_1$ and $S_2$, each connected to two destination nodes, $D_1$ and $D_2$. The routers are divided into 3 clusters as the information provided by the network manager: $C_1$ ($R_1, R_2, R_3$), $C_2$ ($R_4, R_5$) and $C_3$ ($R_6, \ldots$) as shown in Figure 1. The available routes at the network manager for all source-destination pairs are given in Table 1. Please note that this is not the list of all possible routes between

![Fig. 1: An example network scenario](image-url)

**TABLE 1: Available routes between the nodes**

<table>
<thead>
<tr>
<th>(SRC, DEST)</th>
<th>Routes</th>
</tr>
</thead>
<tbody>
<tr>
<td>($S_1, D_1$)</td>
<td>$r_{S_1, D_1}^1 : (R_1, R_2)$, $r_{S_1, D_1}^2 : (R_1, R_3, R_4)$</td>
</tr>
<tr>
<td>($S_1, D_2$)</td>
<td>$r_{S_1, D_2}^1 : (R_1, R_5), r_{S_1, D_2}^2 : (R_1, R_3, R_4)$</td>
</tr>
<tr>
<td>($S_2, D_1$)</td>
<td>$r_{S_2, D_1}^1 : (R_4, R_5, R_3)$, $r_{S_2, D_1}^2 : (R_4, R_5, R_2)$</td>
</tr>
<tr>
<td>($S_2, D_2$)</td>
<td>$r_{S_2, D_2}^1 : (R_4, R_5, R_3)$, $r_{S_2, D_2}^2 : (R_4, R_5, R_2)$</td>
</tr>
</tbody>
</table>

**TABLE 2: BC values of routers**

<table>
<thead>
<tr>
<th>(Routers)</th>
<th>BC value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R_1$</td>
<td>2</td>
</tr>
<tr>
<td>$R_2$</td>
<td>2.5</td>
</tr>
<tr>
<td>$R_3$</td>
<td>3.5</td>
</tr>
<tr>
<td>$R_4$</td>
<td>4</td>
</tr>
<tr>
<td>$R_5$</td>
<td>2</td>
</tr>
<tr>
<td>$R_6$</td>
<td>1</td>
</tr>
</tbody>
</table>

3. BC is originally defined for shortest paths [15], but we define instead of paths available at the network manager. The same is the case with GBC [15].

different source and destination pairs. In this example, we only consider the intermediate routers as objects in the paths. Suppose a large-scale failure has occurred and routers $R_1$ and $R_2$ have failed. We observe the following symptoms: source node $S_1$ is disconnected from both the destination nodes $D_1$ and $D_2$, and source node $S_2$ is disconnected from destination node $D_1$, but is connected to destination node $D_2$.

We obtain the different hypothesis lists given in Table 3 through various algorithms: MAX-COVERAGE (MC), Cluster-MAX-COVERAGE (CMC) with only negative symptoms, and with both negative and positive symptoms. Given the negative symptoms, the BC values for various routers are listed in Table 2. MC iteratively selects the router or routers with maximum BC value, i.e., in our example router $R_3$ is picked first, and then router $R_2$ that has second maximum of all BC values, until all given negative symptoms are explained ($S_1$ is disconnected from the nodes $D_1$ and $D_2$, and $S_2$ is disconnected from the node $D_1$). Therefore, MC localizes only $R_2$, but not $R_1$ which has failed. Hence, $R_1$ is a false negative.

CMC with only negative symptoms selects the cluster to which $R_2$ and $R_3$ belong, i.e., cluster $C_1$ which includes routers $R_1$, $R_2$, and $R_3$. Hence, the hypothesis list of CMC with only negative symptoms will have no false negatives, but contains $R_3$ which is a false positive. Using the information of a positive symptom that says source $S_2$ is connected to $D_2$ in CMC, we can recognize that router $R_3$ is a false positive and eliminate it from the hypothesis list since it is present in both the routes between $S_2$ and $D_2$. This shows why CMC with negative symptoms has fewer false negatives than MC, and the false positives in the hypothesis list can be
Further reduced using positive symptoms.

<table>
<thead>
<tr>
<th>MC</th>
<th>CMC with negative symptoms</th>
<th>CMC with negative and positive symptoms</th>
</tr>
</thead>
<tbody>
<tr>
<td>(R2, R1)</td>
<td>(R1, R2, R3)</td>
<td>(R1, R2)</td>
</tr>
</tbody>
</table>

**TABLE 3**: Hypothesis lists for various algorithms

## 5 Cluster MAX-COVERAGE

CMC localizes large-scale failures in networks given end-to-end observed symptoms and the paths among different nodes that are collected from reporting nodes. There are three phases in CMC. The first phase uses a greedy approach and a clustering model to generate a hypothesis list of objects using only negative symptoms. This is called the Hypothesis generation phase. In the second phase, we remove some of the extra objects (false positives) in the hypothesis list by identifying impossible situations using positive symptoms in the network; this is called the DEC\_FP phase. Finally in the third phase, we divide the objects in the resulting hypothesis list into clusters and rank them. We call this the Ranking phase.

### 5.1 Clustering Model

The network may be divided into different clusters based on various attributes of the network. For example, objects in the same building or geographical area may be considered to be in one cluster. Given the clustered failures, it is most likely that all objects in a given cluster may be either faulty or not faulty, collectively.

In our clustering model, we assume that all objects in the network are partitioned into clusters and complete information about these clusters is known a priori to the network manager. Our model is generic since the network can be divided into clusters based on any general attribute chosen by the network manager. As explained in Section 4.1, there are c clusters in total and the number of objects in each cluster can vary. Cluster $C_i$ with k arbitrary objects is represented by $C_i = \{c_1^i, c_2^i, ..., c_k^i\}$. In addition, since clusters need not be mutually exclusive, a single object can be present in multiple clusters. Alternatively, we can also consider that objects within a certain distance metric from each other comprise a cluster if complete cluster information is not available to the network manager [11].

### 5.2 Hypothesis Generation Phase

The hypothesis generation phase of CMC is inspired by MC, described in Section 3. The second step of MC is modified in CMC enabling the latter to diagnose large-scale failures. The pseudo-code for this phase is given in Figure 1 in the supplemental file [12]. We pick a cluster of objects rather than a single object, while employing a greedy approach. To be precise, in CMC we choose a cluster of objects that are present around the object with the maximum BC (selObjects in Line 14). This is given in Line 15-16 of the pseudo-code. In this way, CMC iteratively selects the network objects until all negative symptoms are accounted for. Therefore, in large-scale failure scenarios, CMC diagnoses more failed objects than MC does.

CMC deals with the issue of multiple paths between different nodes in the network, with the help of different topology snapshots (TopologySnapshots) during a given time interval of failure. These snapshots reflect the routing changes that have occurred during that interval. This phase is applied to each snapshot to output different hypothesis lists. CMC takes the union of these hypothesis lists to output one hypothesis list (unionHypList). MC also tackles the issue of multiple paths in a similar way.

In large-scale failures, CMC results in fewer false negatives compared to MC. However, the hypothesis list may have a high number of false positives, since clusters are chosen instead of a single object. This is more desirable than a high number of false negatives, because in fault management we can probe and confirm false positives, unlike false negatives. To reduce the cost of these additional probes, we propose a novel scheme to eliminate false positives from the hypothesis list of objects with the help of observed positive symptoms.

### 5.3 DEC\_FP Phase

In the DEC\_FP phase, we decrease the number of false positives in the hypothesis list of possible failed objects. Unlike other works in the area of fault diagnosis [2], [3], we propose a novel combinatorial scheme with the help of positive symptoms. This is different from netCSI [10] that employs a combinatorial approach on various combinations of all objects, which are consistent with both positive and negative symptoms.

We employ an algorithm to find the most likely false positives in the DEC\_FP phase; we call our approach the combinatorial algorithm with positive symptoms (COMB\_PS). The algorithm considers only the objects in the list that is obtained from the hypothesis generation phase, and employs a combinatorial approach to find the combinations of objects in the hypothesis list that are inconsistent with positive symptoms. The details of the COMB\_PS algorithm is given in Section 1 of supplemental file [12], and the pseudo-code of this algorithm is given in Figure 2 of the supplemental file [12]. This strategy may also be used for independent failures.

### 5.4 Ranking Phase

We introduce a ranking scheme for more efficient diagnosis of large-scale failures. The aim of our ranking scheme is to reduce the false positives by outputting a list of the ranked clusters of failed objects. We do this because of

---

1045-9219 (c) 2013 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
the premise that the objects in the same cluster will fail together with a higher likelihood. The ranking scheme is based on BC values of objects and the previously given clustering model (Sections 4.2, 5.1). Our scheme assigns the highest rank to a cluster of an object that has the maximum BC value (rankObj) and the rest of the objects in the hypothesis list that belong to the same cluster (rankHypList). The clusters are picked according to the given clustering model. We continue this process until all the objects in the hypothesis list are exhausted and ordered in clusters. Therefore, in the ranking phase, we are essentially dividing the objects in the hypothesis list into different clusters, and then ranking them. The pseudo-code of the ranking phase is shown in Figure 3 of the supplemental file [12].

Once we have these ranked clusters, we select the clusters which have a lower rank than a threshold and group them into a new truncated hypothesis list. This threshold rank can be chosen by the network manager, and is called the cumulative rank. The resulting truncated hypothesis list will have fewer false positives when compared to the original hypothesis list. Therefore, false positives are further reduced in this phase. We describe the advantages of cumulative rank and evaluate our ranking scheme in Sections 6.2 and 6.3.2.

6 Evaluation of CMC

6.1 Simulation setup

We evaluate the performance of CMC for two different types of topologies, random and realistic. We run simulations with random topology networks that are generated by placing 100 nodes at random locations in a 2-dimensional space. The realistic topology is taken from Rocketfuel trace data provided by the University of Washington [16]. A network with 315 backbone routers that are connected by 972 links in an ISP (Sprint in US) is used for our simulations. Note that we deal with large-scale failures due to highly unexpected events such as natural disasters or WMD attacks, therefore it is tough to get real data corresponding to failure spots. Hence, we simulated large-scale failures synthetically and used realistic topologies for simulations.

In both networks, multiple paths are generated among all nodes using Dijkstra’s algorithm. We generate these paths in multiple instances by failing certain nodes in each instance. The number of multiple paths between any two nodes should be at least a given value (5 is the value in our simulations). Without loss of generality, we only consider links as objects in a path, while collecting the path information. In every simulation, a certain number of reporting nodes is selected randomly out of all nodes to report end-to-end symptoms. We ran 15 trials in each simulation and present the aggregated results over all the trials.

For simulation purposes, clustering is done in different ways for networks with random and realistic topologies. In random networks, the network is divided into different disjoint clusters using a k-means clustering algorithm. With the number of clusters (k) as the input, the k-means clustering algorithm generates k clusters, each with a variable number of nodes. For the network with realistic topology, all the nodes in a particular city are regarded as one cluster; there are 24 clusters in total.

We simulate two types of failure modes: random cluster failure and random space failure. In the random cluster failure mode, we choose a random cluster out of all the clusters given in the network and fail all objects in that particular cluster. This failure mode mimics a realistic scenario where all objects in a building (considered as one cluster) may fail due to various reasons. For random space failure mode, a circle with a certain radius v is randomly chosen, and all objects within that circle are failed. This scenario is similar to a case where objects that are geographically adjacent in multiple clusters can fail together. Moreover, the failure modes are different from the information in clustering model that is given in Section 5.1.

In Section 6.3 we evaluate and compare the following algorithms: CMC with only the first phase, i.e., with no positive symptoms (CMC_NO_POS), CMC with both first and second phases (CMC_POS), MC, and netCSI.

6.2 Performance Metrics

CMC outputs a hypothesis list that consists of objects which have possibly failed. In this context, we define three metrics which we compare to the ground truth during large-scale failures: false negatives, false positives, and accuracy. False negatives (FNs) are defined as the fraction of objects in the ground truth that are not present in the hypothesis list. False positives (FPs) are defined as the fraction of objects in the hypothesis list which are not part of the ground truth. Accuracy is the complement of false negatives, i.e., fraction of objects in the ground truth that are diagnosed correctly. Since we use a ranking scheme in CMC, we define these terms for a hypothesis list with various ranked clusters.

To decrease the false positives, we order our hypothesis list of objects through clustering of objects and then ranking the clusters. In this context, we assume that the network manager chooses a cumulative rank (CR) to truncate the hypothesis list by selecting only a few highly ranked clusters. We compare the performance of different cumulative ranks against the case in which the ranking phase is not employed.

6.3 Results

6.3.1 Performance of CMC

To evaluate and compare the performance of the algorithms, we simulate random cluster failure mode. We present results for networks with random topologies with the number of reporting nodes = 20. We consider three instances of k (number of clusters = 40, 30, 20).

As shown in Figure 2(a), CMC_NO_POS results in significantly fewer false negatives than MC, but incurs
more false positives as shown in Figure 2(b). CMC_POS reduces the false positives by 2-fold, while the increase in false negatives is negligible comparatively. This shows the effectiveness of including positive symptoms in failure diagnosis. We observe similar trends in the network with realistic topology, but with high gains because of varied degree distribution compared to random topology; few nodes in realistic topology have a very high betweenness centrality compared to other nodes. The results are shown in Section 1.3 of the supplemental file [12].

We run simulations to choose the appropriate value for the parameter \(\text{obj}_{th}\) by evaluating the performance of CMC in different scenarios. From the simulations, we determine the appropriate value as \(\text{obj}_{th}=3\) for CMC_POS. The details of these simulation results are given in Section 1.4 in the supplemental file [12].

### 6.3.2 Ranking Phase

The ranking phase can be employed with both CMC_NO_POS and CMC_POS. We show results for a random topology with 20 clusters in Figure 3. We compare the following cases: \(\text{no ranking}\) (i.e., ranking is not employed), \(CR=1\), \(CR=2\), and \(CR=3\). As shown in Figure 3, \(CR=1\) has same FNs as \(\text{no ranking}\), but FPs are much lower than in the case \(\text{no ranking}\). This shows that the ranking scheme can diagnose the same number of failed objects as the scheme without ranking, but with fewer false positives. In our simulations, we observe that the top rank cluster finds all the failed objects more often than not, because the failure mode and the cluster model are consistent, and also only one cluster fails. However, in general cluster failure settings, \(CR\) with a very low value (instead of 1) that is picked by the network manager may achieve better performance.

### 6.3.3 Varying Reporting Nodes

In this section, we focus on the performance of various algorithms while varying the number of reporting nodes. The accuracy of CMC_POS, CMC_NO_POS, MC, and netCSI are evaluated for both random and realistic topologies. We present results for the realistic topology in this section, and the results for the random topology are given in Section 1.5 of the supplemental file [12]. The accuracy of fault diagnosis increases with an increase in the number of reporting nodes. In a network of 315 nodes with a realistic topology, we observe high gains, i.e., with 25 reporting nodes, CMC_NO_POS and CMC_POS achieve an accuracy of almost 100%. However, in the case of MC, it requires 275 reporting nodes to accomplish the same level of accuracy. Hence, a higher number of reporting nodes results in a higher burden on the network manager while collecting symptoms and paths, in terms of network overhead and probing costs. Therefore, both CMC_NO_POS and CMC_POS achieve 100% accuracy with very low overhead when compared to MC.

netCSI achieves almost the same accuracy as both CMC_NO_POS and CMC_POS, but incurs a high runtime, since it employs a combinatorial approach instead of a greedy approach. We observe this in our simulation results which are provided in Section 1.6 of the supplemental file [12].

### 6.3.4 Partial Symptoms at a Reporting Node

In Section 6.3.3, we described the effect of incomplete symptoms on CMC by varying the number of reporting nodes under the assumption that each reporting node has complete information of symptoms. Here, we address another kind of incompleteness in information that results from partial knowledge of symptoms at a selected reporting node. This may occur because every reporting node may not be able to determine all symptoms quickly. If the connectivity status of a destination node corresponding to a reporting node is not known, then no symptom is reported.

We study the impact of this kind of incomplete information on our CMC algorithm and present the results for the performance of CMC when compared to other algorithms.

---

**Fig. 2:** Random cluster failures in random networks

**Fig. 3:** Performance of Ranking Scheme

**Fig. 4:** Accuracy vs reporting nodes for realistic topology
We compare the performance of CMC with MC and netCSI under different varieties of partial symptoms. We consider four different cases of partial symptom information. The first case is where the number of symptoms at each reporting node falls between 30% and 90% of the total number of both positive and negative symptoms, and in the remaining three cases every chosen reporting node has 90%, 60% or 30% of total number of both positive and negative symptoms.

We consider a random network topology of 100 nodes, and trigger random space failures as large-scale failures. Each simulation is run for 10 trials. In Figures 5 and 6, we compare CMC separately with MC and netCSI for the above four cases under a varying number of reporting nodes. We do not show the results with 60% of partial information for clarity in the graphs.

CMC has much better accuracy than both MC and netCSI in all the cases, especially when there are fewer reporting nodes in the network. In particular, CMC with 30% partial information achieves 100% accuracy with 40 reporting nodes, whereas neither MC nor netCSI achieve 100% accuracy even with 100 reporting nodes as shown in Figures 5 and 6.

In Figure 5, we see that CMC performs much better than MC for all the cases of partial information with varying number of reporting nodes. As shown in Figure 6, CMC also performs much better than netCSI when there are fewer reporting nodes (≤ 50 reporting nodes) for all the cases. This is important because not every node can report its symptoms to the network manager under large-scale failures.

In addition to comparing various algorithms, we investigate the individual significance of diverse symptoms. We observe that for CMC, reporting nodes require fewer negative symptoms and a substantial amount of positive symptoms to diagnose failures with fewer false positives and fewer false negatives (complement of accuracy). We provide the details of these simulations in Section 1.7 of the supplemental file [12].

We provide a discussion of our results in Section 1.8 of the supplemental file [12].

7 Adaptive-Max-Coverage

As our results in Section 6.3 have shown, CMC outperforms MC for clustered failures. However, in the case of independent failures, MC outperforms CMC in terms of false positives, as shown in Table 4 (Fraction of FPs and FNs). These results are produced using the experimental setup given in Section 6.1. For clustered failures, we simulated random cluster failure mode (Section 6.1), and for independent failures, we failed links that are selected randomly.

Therefore, to handle the case in which there may be both independent and clustered failures, we propose a hybrid fault diagnosis approach called Adaptive MAX-COVERAGE (AMC). The key element of AMC is determining whether a failure is independent or clustered, so that an appropriate fault diagnosis algorithm can be employed accordingly. AMC classifies failure types with the help of the Bayes decision technique that makes use of the negative symptoms to differentiate various types of failures.

7.1 Basic Approach

Our decision technique uses the classic Bayesian decision theory to classify different failure types. We consider two classes of failures: independent and clustered failures. We define two events $IF$ and $CF$, for independent and clustered failures, respectively. In addition, we define a random variable, $X$, to represent the number of negative symptoms in the network due to a failure. Given the probability mass functions of $X$ for both independent and clustered failures (prior to the current failure), and the observed negative symptoms due to a current failure in real-time (ns), we use Bayesian rule to compute the posterior failure probability. The following equations explain our technique.
Here, note that to determine the negative symptoms, 

\[ P(IF|X = ns) = \frac{P(X = ns|IF)P(IF)}{P(X = ns|IF)P(IF) + P(X = ns|CF)P(CF)} \] (2)

\[ P(CF|X = ns) = \frac{P(X = ns|CF)P(CF)}{P(X = ns|IF)P(IF) + P(X = ns|CF)P(CF)} \] (3)

\[ P(CF|X = ns) \geq P(IF|X = ns): \text{Clustered failure} \]

\[ P(CF|X = ns) < P(IF|X = ns): \text{Independent failure} \] (4)

where \( P(IF) \) and \( P(CF) \) are the prior probabilities of independent and clustered failure events.

The values of \( P(X = ns|IF) \) and \( P(X = ns|CF) \) are determined using the probability mass functions of \( X \) given independent and clustered failure events, i.e., \( X|IF \) and \( X|CF \). We explain our methodology to evaluate these probability mass functions using a database in Section 7.2. Finally, we use the the posterior probabilities, \( P(CF|X = ns) \) and \( P(IF|X = ns) \), which are determined with the help of Bayes rule (Equations (2) and (3)), to classify failures as shown in Equation (4). To enable efficient and robust fault diagnosis, we extend Bayes decision technique using a cost function that represents penalties of wrong decisions. The details of the cost function is given in Section 2.1 of the supplemental file [12]. In addition, we provide the details of prior probabilities, \( P(IF) \) and \( P(CF) \), in Section 2.2 of the supplemental file [12].

### 7.2 Failure Data

We utilize a database of failure data that consists of both offline and online data about independent and clustered failures. The database contains the number of end-to-end negative symptoms in the network for both types of failures. This helps us to estimate the probability mass functions of random variables, \( X|IF \) and \( X|CF \). Offline data are collected from samples of both independent and clustered failures, chosen randomly in the network. The online data are obtained from the feedback given to the network manager when the failures occur in real-time.

For offline data, we use the network measures \( BC \) and \( GBC \), to determine the end-to-end negative symptoms, instead of enumerating the symptoms by simulating failures. In the single object failure case, the number of negative symptoms is equal to \( BC \) of the failed object. When multiple objects fail simultaneously and independently, the total number of end-to-end negative symptoms is estimated as the sum of the \( BC \) values of the failed objects. For clustered failures, we choose samples that include both random cluster and random space (Section 6.1). For both modes, the number of end-to-end negative symptoms is equal to the cluster’s \( GBC \) value.

Here, note that to determine the negative symptoms, we are not simulating failures offline which involves high complexity, but instead we are using relevant graph measures.

This offline data enables us to construct the histograms that represent the frequency of number of negative symptoms corresponding to both independent and clustered failures. These histograms are updated with online data after each failure, i.e., using the number of negative symptoms observed in the network whenever a failure occurs in real-time. We use a well-known technique called kernel density estimation [17] to estimate the probability mass function of \( X|IF \) and \( X|CF \).

### 8 Evaluation of AMC

We use the same experimental setup given in Section 6.1 to evaluate AMC. Here, we simulate a series of failures that include both independent and clustered failures. We evaluate AMC with the following parameters: number of reporting nodes (\( R \)), and the probability of independent failures (\( P \)). The number of reporting nodes signifies the extent of partial information at the network manager, whereas probability of independent failures specifies how often independent failures occur in a series of failures. We use the fraction of false positives and false negatives (FPs and FNs) that are defined in Section 6.2 as metrics.

We evaluate and compare various fault diagnosis algorithms for three cases. The values of the variables for different cases are given in Table 5. In CASE-1, every variable has a base value. In the next two cases, we vary only one variable in each case as shown in Table 5. We do this to understand how different variables affect the various algorithms individually. We give results for a 100 node network with random topology.

Figure 7 shows that AMC and CMC algorithms produce fewer FNs than MC in all cases. This is because of the large-scale clustered failures that are present in the simulated series of failures. However, FPs in the case of CMC are high when compared to other algorithms as shown in Figure 8, due to the degradation of its performance during independent failures. The AMC algorithms diagnose the failures efficiently, i.e., resulting in a good number for both FPs and FNs.

In CASE-2, the number of reporting nodes is high, i.e., there is more information at the network manager. Both FPs and FNs decrease as shown in Figures 8 and 7 for MC and AMC algorithms. For CMC, the number of FPs remains high because of performance degradation during independent failures. In CASE-3, unlike CASE-1, clustered failures occur more often than independent failures. We observe that FNs increase for MC, and FPs decrease for CMC, when compared to CASE-1.

We next vary the maximum number of independent failures, i.e., maximum number of objects that can fail at a time during independent failures. We observe that the FPs and FNs do not change much for any of the algorithms, especially for AMC algorithms, i.e., the classification success rates (success rate of our Bayes

<table>
<thead>
<tr>
<th>CASE-1</th>
<th>CASE-2</th>
<th>CASE-3</th>
</tr>
</thead>
<tbody>
<tr>
<td>( R=20 )</td>
<td>( R=50 )</td>
<td>( R=20 )</td>
</tr>
<tr>
<td>( P=0.8 )</td>
<td>( P=0.8 )</td>
<td>( P=0.2 )</td>
</tr>
</tbody>
</table>

### TABLE 5: Values of variables in different cases
decision technique) do not change considerably with an increase in the maximum number of independent failures. Furthermore, when we change the connectivity density of the network and hence when the size of clustered failures is increased, FPs and FNs do not vary much for different algorithms, except for MC. This is because MC treats all clustered failures as independent failures.

**Fig. 7: Evaluation of AMC: False Negatives**

**Fig. 8: Evaluation of AMC: False Positives**

We run simulations that evaluate AMC with a cost function for different values of $\alpha$. Results of these simulations reveal that the cost function gives the network manager more control over relative emphasis on FPs and FNs in failure diagnosis. The details of the results are given in Section 2.3 of the supplemental file [12].

9 Conclusion

We introduced CMC, an algorithm to diagnose faults due to large-scale failures accurately and quickly, in spite of having incomplete symptom information. CMC accomplishes 100% accuracy with half of the reporting nodes in a random topology and one-tenth of the reporting nodes in a realistic topology, when compared to MC. This gain is achieved much faster than netCSI—sometimes by two orders of magnitude. CMC utilizes positive symptoms to output a hypotheses list that has 45% fewer false positives when compared to the case without any positive symptoms. CMC performs much better than both netCSI and MC when there are fewer negative and positive symptoms at a chosen reporting node. In addition, we presented results that shows the diversifying nature of positive and negative symptoms at a reporting node. We also propose an adaptive algorithm called AMC to diagnose both independent and clustered failures effectively with the help of Bayesian decision technique. AMC produces a hypothesis list with a low number of false positives and false negatives when compared to CMC and MC over a series of failures.

Acknowledgements

We acknowledge the inputs of Daniel Kifer. This research was sponsored by the U.S. Army Research Laboratory and the U.K. Ministry of Defence and was accomplished under Agreement Number W911NF-06-3-0001. The views and conclusions contained in this document are those of the author(s) and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Army Research Laboratory, the U.S. Government, the U.K. Ministry of Defence or the U.K. Government. The U.S. and U.K. Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon. This work was supported in part by the Defense Threat Reduction Agency under grant HDTRA1-10-1-0085.

References

Srikar Tati

Srikar Tati is a graduate student in PhD program at Pennsylvania State University. He received the Bachelors of technology (B. Tech Hons.) degree at Indian Institute of Technology, Kharagpur in 2006 and Masters of Sciences (MS) degree at Pennsylvania State University in 2008. His research interests are in broad areas of design, modeling and analysis of network systems, large distributed systems, wireless networks etc. As part of his dissertation, he worked on various problems that arise due to challenges in failure and performance diagnosis algorithms in computer networks. He previously worked on few problems in wireless networks. He published his work in conferences such as IEEE DSN, ICNP, SRDS etc. He reviewed papers for JSAC network science, IEEE Globecom etc. He is a member of IEEE and ACM.

Bongjun Ko

Bong Jun Ko received B.S. and M.S. degrees in electrical engineering from Seoul National University, Korea, and received the Ph.D. degree in electrical engineering at Columbia University, New York. He has worked as a senior member of research staff in Philips Research North America, as a research engineer at LG Electronics in Korea, and has co-founded NeoMTel, Korea. He is currently a research staff member in IBM T. J. Watson Research Center, working in Networking Technology Area, and has research interests in modeling, analysis, and design of large distributed systems, mobile and wireless networks. He is the recipient of the best paper awards in IEEE ICNP 2003 and in IEEE/IFIP IM 2013.

Guohong Cao

Guohong Cao received the BS degree in computer science from Xian Jiaotong University and received the PhD degree in computer science from the Ohio State University in 1999. Since then, he has been with the Department of Computer Science and Engineering at the Pennsylvania State University, where he is currently a Professor. He has published more than 200 papers in the areas of wireless networks, wireless security, vehicular networks, wireless sensor networks, cache management, and distributed fault tolerant computing. He has served on the editorial board of IEEE Transactions on Mobile Computing, IEEE Transactions on Wireless Communications, IEEE Transactions on Vehicular Technology, and has served on the organizing and technical program committees of many conferences, including the TPC Chair/Co-Chair of IEEE SRDS’2009, MASS’2010, and INFOCOM’2013. He was a recipient of the NSF CAREER award in 2001. He is a Fellow of the IEEE.

Ananthram Swami

Ananthram Swami (F’08) received the B.Tech. degree from IIT-Bombay; the M.S. degree from Rice University, and the Ph.D. degree from the University of Southern California (USC), all in Electrical Engineering. He has held positions with Unocal Corporation, USC, CS-3 and Malgudi Systems. He was a Statistical Consultant to the California Lottery, developed a Matlab-based toolbox for non-Gaussian signal processing, and has held visiting faculty positions at INP, Toulouse. He is with the US Army Research Laboratory (ARL) where he is the Army’s ST for Network Science. His work is in the broad area of network science, with emphasis on wireless communication networks. He is an ARL Fellow and Fellow of the IEEE. He is a member of the Founding Steering Committee of the IEEE Transactions on Network Science and Engineering; has served as guest editor for multiple special issues (SI), most recently IEEE JSAC SI on Network Science; has co-organized five workshops, most recently IEEE SPAWC’10 and was TPC co-chair for USIPCO'13. He was a tutorial speaker on ‘Networking Cognitive Radios for Dynamic Spectrum Access’ at ICASSP 2008, DySpan 2008, MILCOM 2008, and ICC 2010, co-editor of the 2007 Wiley book “Wireless Sensor Networks: Signal Processing & Communications Perspectives”; and recipient of best conference paper award at IEEE Trustcom 2009 and IEEE ICDCS 2013.

Thomas F. La Porta

Thomas F. La Porta is the William E. Leonhard Chair Professor in the Computer Science and Engineering Department at Penn State. He received his B.S.E.E. and M.S.E.E. degrees from The Cooper Union, New York, NY, and his Ph.D. degree in Electrical Engineering from Columbia University, New York, NY. He joined Penn State in 2002. He is the Director of the Institute of Networking and Security Research at Penn State. Prior to joining Penn State, Dr. La Porta was with Bell Laboratories since 1986. He was the Director of the Mobile Networking Research Department in Bell Laboratories, Lucent Technologies where he led various projects in wireless and mobile networking. He is an IEEE Fellow, Bell Labs Fellow, received the Bell Labs Distinguished Technical Staff Award in 1996, and an Eta Kappa Nu Outstanding Young Electrical Engineer Award in 1996. He also won a Thomas Alva Edison Patent Awards in 2005 and 2009. Dr. La Porta was the founding Editor-in-Chief of the IEEE Transactions on Mobile Computing, and currently serves on its Steering Committee (he was Chair from 2009-2010). He served as Editor-in-Chief of IEEE Personal Communications Magazine. He was the Director of Magazines for the IEEE Communications Society and was on its Board of Governors for three years. He has published numerous papers and holds 35 patents.