RC23698 (W0508-102) August 18, 2005
Computer Science

IBM Research Report

Adaptive Diagnosis in Distributed Systems

Irina Rish, Mark Brodie, Sheng Ma, Natalia Odintsova, Alina Beygelzimer, Genady Grabarnik
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 704
Yorktown Heights, NY 10598

Karina Hernandez
IBM Systems and Technology Group
11501 Burnet Road
906-3014E
Austin, TX 78758

Research Division
Almaden - Austin - Beijing - Haifa - India - T. J. Watson - Tokyo - Zurich

Adaptive Diagnosis in Distributed Systems

Irina Rish, Mark Brodie, Sheng Ma, Natalia Odintsova,
Alina Beygelzimer, Genady Grabarnik, Karina Hernandez

***Abstract*— Real-time problem diagnosis in large distributed computer systems and networks is a challenging task that requires fast and accurate inferences from potentially huge data volumes. In this paper, we propose a cost-efficient, adaptive diagnostic technique called *active probing*. Probes are end-to-end test transactions that collect information about the performance of a distributed system. Active probing uses probabilistic reasoning techniques combined with information-theoretic approach, and allows a fast online inference about the current system state via active selection of only a small number of most-informative tests. We demonstrate empirically that the active probing scheme greatly reduces both the number of probes (from 60% to 75% in most of our real-life applications), and the time needed for localizing the problem when compared with non-adaptive (pre-planned) probing schemes. We also provide some theoretical results on the complexity of probe selection, and the effect of "noisy" probes on the accuracy of diagnosis. Finally, we discuss how to model the system's dynamics using Dynamic Bayesian networks, and an efficient approximate approach called sequential multifault; empirical results demonstrate clear advantage of such approaches over "static" techniques that do not handle system's changes.**

***Index Terms*— Diagnosis, probabilistic inference, Bayesian networks, information gain, computer networks, distributed systems, end-to-end transactions.**

I. Introduction

Accurate diagnosis and prediction of unobserved states of a large, complex, multi-component system by making inferences based on the results of various tests and measurements is a common problem occurring in practice. Numerous examples include medical diagnosis, airplane failure isolation, systems management, error-correcting coding, and speech recognition. Achieving high diagnostic accuracy may require performing a large number of tests, which can be quite expensive. It is therefore essential to improve scalability and cost-efficiency of diagnosis by using only the most relevant measurements at any time point, i.e. by making inference more adaptive ("context-specific") to the current system state and observations.

The key component of the approach proposed in this paper is an adaptive measurement technique, called *active probing*[1], that allows a fast online inference about the current system state via active selection of only a small number of most-informative measurements called *probes*. A probe is a test transaction whose outcome depends on some of the system's components; accurate diagnosis can be achieved by appropriately selecting the probes and analyzing the probe outcomes. Our main contribution is in providing a theoretical foundation and a set of practical techniques for implementing efficient probing strategies.

Although our methods are quite generic and are applicable to a wide variety of problem areas, we will focus specifically on the area of distributed systems management. The rapid growth in size and complexity of distributed systems makes performance management tasks such as problem diagnosis – detecting system problems and isolating their root causes – an increasingly important but also extremely difficult task. For example, in IP network management, we would like to quickly identify which router or link has a problem when a failure or performance degradation occurs in the network. In the e-Commerce context, our objective could be to trace the root-cause of unsuccessful or slow user transactions (e.g. purchase requests sent through a web server) in order to identify whether it is a network problem, a web or back-end database server problem, etc. Another example is real-time monitoring, diagnosis and prediction of the "health" of a large cluster system containing hundreds or thousands of workstations performing distributed computations (e.g., Linux clusters or GRID-computing systems).

A. Current Problem-Diagnosis Approaches

A commonly used approach to problem diagnosis in distributed systems management is *event correlation* [5], [6], [7], in which every managed device is instrumented to emit an alarm when its status changes. By correlating the received alarms a centralized manager is able to identify the problem. However, this approach usually requires heavy instrumentation, since each device needs to have the ability to send out the appropriate alarms. Also, it may be difficult to ensure that alarms are sent out, e.g. by a device that is down. Finally, it might be impossible to obtain the event data from all parts of the network, especially if it contains "black boxes" such as proprietary components.

To avoid these problems, an alternative diagnostic approach has been developed that is based on *end-to-end probing* technology [8], [9], [10]. A *probe* is a test transaction whose outcome depends on some of the system's components; diagnosis is performed by appropriately selecting the probes and analyzing the results. In the context of distributed systems, a probe is a program that executes on a particular machine (called a probe station) by sending a command or transaction to a server or network element and measuring the response. The *ping* and *traceroute* commands are probably the most popular probing tools that can be used to detect network availability. Other probing tools, such as IBM's EPP technology ([8]), provide more sophisticated, application-level probes.

I. Rish, M.Brodie, S. Ma, N. Odintsova, A. Beygelzimer, G. Grabarnik are with the IBM T.J. Watson Research Center, 19 Skyline Drive, Hawthorne, NY 10532

K. Hernandez is with the IBM Systems and Technology Group, 11501 Burnet Road, 906-3014E, Austin TX 78758

[1]This paper summarizes and extends preliminary results presented by the authors in several conference papers, such as [1], [2], [3], [4].

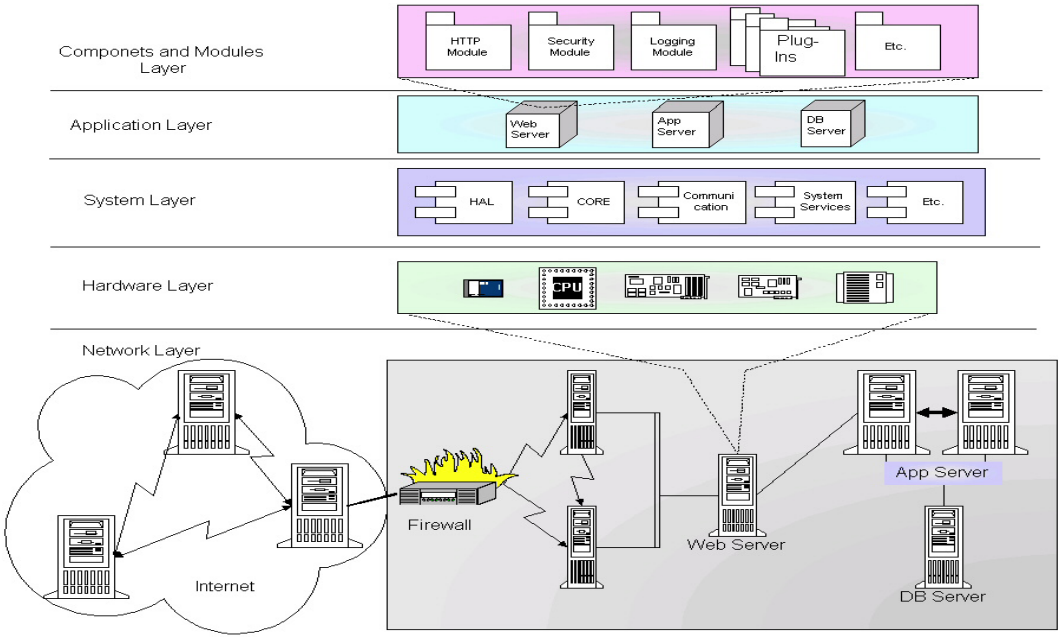


Fig. 1. Illustrative example where probing can be used at multiple levels.

For example, probes can be sent in the form of test e-mail messages, web-access requests, a database query, and so on.

Figure 1 illustrates the core ideas of probing technology. The bottom left of the picture represents an external cloud (e.g. the Internet), while the greyed box in the bottom middle and right represents an example intranet - e.g. a web site hosting system containing a firewall, routers, web server, application server running on a couple of load balanced boxes, and database server. Each of these contains further substructure - the figure illustrates the various layers underlying the components. Probing can take place at multiple levels of granularity; the appropriate choice of granularity depends on the task probing is used for. For example, to test a Service Level Agreement (SLA) stating response time one need only probe one point of Figure 1, the point of contact of the external cloud and the intranet. In order to find more detailed information about the system one could probe all network segments as well the web server, application server, database server - all the elements of the intranet in the network layer. If we need to do problem determination or tune up the system for better performance, we may also need to consider more detailed information; e.g. from the system layer (some systems allow instrumentation to get precise information about system components) or component and modules layer, and so on. For each task appropriate probes must be selected and sent and the results analyzed.

In practice, probe planning (i.e., choice of probe stations, targets and particular transactions) is often done in an ad-hoc manner, largely based on previous experience and rules-of-thumb. Thus, it is not necessarily optimized with respect to probing costs (related to the number of probes and probe stations) and diagnostic capability of a probe set. More recent work [9], [10] on probing-based diagnosis focused on optimizing the probe set selection and provided simple heuristic search techniques that yield close-to-optimal solutions. However, the existing probing technology still suffers from various limitations:

1) Probes are selected off-line (*pre-planned probing*), and run periodically using a fixed schedule (typically, every 5 to 15 min). For diagnostic purposes, this approach can be quite inefficient: it needs to construct and run repeatedly an unnecessarily large set of probes capable of diagnosing *all* possible problems, many of which might in fact never occur. Further, when a problem occurs, there may be a considerable delay in obtaining all information necessary for diagnosis of this particular problem. Thus, a more adaptive probe selection is necessary.
2) Another limitation of existing techniques, including both event correlation [7] and probing [9], [10], is their non-incremental (“batch”) processing of observed symptoms (alarms, events, or probes), which is not quite suitable for continuous monitoring and real-time diagnosis. An incremental approach is required that continuously updates the current diagnosis as more observations become available.
3) Finally, existing approaches typically assume a static model of the system (i.e., the system state does not change during diagnostic process). While “hard” failures of components are indeed relatively rare, “soft” failures such as performance degradations (e.g., response time exceeding certain threshold) may happen more frequently; in a highly dynamic system ‘failure” and “repair” times may get comparable with the average diagnosis time. This can lead to erroneous interpretation of some contradictory observations as “noise” [7] when in fact they may indicate changes in the system. A more sophisticated model that accounts for system dynamics can provide a more accurate diagnosis in such cases.

B. Our Contributions

In this paper, we aim at improving the current state-of-art in problem diagnosis by introducing a more adaptive and cost-efficient technique, called *active probing*, that is based on information theory. Combining probabilistic inference with active probing yields an adaptive diagnostic engine that "asks the right questions at the right time", i.e. dynamically selects probes that provide maximum *information gain* about the current system state. We approach diagnosis problem as the task of reducing the uncertainty about the current system state $\mathbf{X}$ (i.e., reducing the entropy $H(\mathbf{X})$) by acquiring more information from the probes, or tests $\mathbf{T}$. Active probing repeatedly selects the next most-informative probe T_k that maximizes the information gain $I(\mathbf{X}; T_k|T_1, ..., T_{k-1})$ given the previous probe observations $T_1, ..., T_{k-1}$. Probabilistic inference in Bayesian networks is used to update the current *belief* about the state of the system $P(\mathbf{X})$.

Active probing is an incremental approach that is well-suited for real-time monitoring and diagnosis. Moreover, it avoids the waste inherent in the pre-planned approach since active probing always selects and sends probes as needed in response to problems that actually occur. Also, active probes are only sent few times in order to diagnose the current problem and can be stopped once the diagnosis is complete. Only a relatively small number of probes needed for fault *detection* should circulate regularly in the network. In practice, active probing requires on average much less probes than the pre-planned probing; for example, it reduced probe set size by up to 75% both in simulated problems and in several practical applications we considered.

Clearly, active probing can be also combined with traditional approaches such as various event correlation techniques [7], [5], [6]. Our diagnostic engine, that uses probabilistic inference in Bayesian networks, can accept any input events, including probes, alarms and other messages, and performs appropriate inferences about the current system state. However, we add the ability of active measurement selection on top of such 'passive' inference capabilities.

In summary, this paper makes the following contributions:
1. We provide a theoretical analysis of optimal probe selection for *fault detection* and *fault diagnosis*, and show that both problems are NP-hard.
2. We reformulate and generalize previously proposed pre-planned probe-selection approach of [9], [10] using information-theoretic framework.
3. We develop an algorithm for active probing and demonstrate its advantages over pre-planned probing (up to 75% savings).
4. We discuss a simple approximation algorithm for diagnosis used when the exact probabilistic inference is intractable, and provide theoretical guarantees on its diagnostic quality in the presence of noise in probe outcomes.
5. Finally, we discuss how to model the system's dynamics using Dynamic Bayesian networks, and an efficient approximate approach called sequential multifault; empirical results demonstrate clear advantage of such approaches over "static" techniques that do not handle system's changes.

The outline of the paper is as follows. Section II provides the basic framework and notation. Section III give a high-level overview of the proposed approach. Section IV discusses pre-planned probe-selection for fault detection and diagnosis. We prove that the optimal probe set selection problem is NP-hard (even for single-fault diagnosis), and briefly describe linear and quadratic-time approximation algorithms proposed in [9], reformulated here in a unifying information-theoretical framework. Section V presents the *active* probing algorithm. In Section VI we focus analysis of probe results using probabilistic inference; we discuss both simple approach based on k-fault assumption and generic multi-fault approach; we also provide theoretical guarantees on diagnostic quality of a simple approximate inference algorithm. Section VIII-B.1 presents empirical results demonstrating advantages of adaptive versus non-adaptive probing on both simulated and real-life problems. Section VIII reports our approach to handling dynamically changing systems, that includes general framework of Dynamic Bayesian Networks (DBNs) and an approximate but more computationally efficient approach called sequential multifault. Finally, section IX describes the architecture and applications of our proof-of-concept system that implements the algorithms described above and functions in a realistic environment. Related work is discussed in Section X, while Section XI provides a summary and describes directions of future work.

II. Definitions and Framework

Let us assume there is a set of system *components*, or *nodes* $\mathbf{N} = (N_1, ..., N_n)$, each of which can be either be "OK" (functioning correctly) or "faulty" (functioning incorrectly). In a distributed system, the nodes may be physical entities such as routers, servers, and links, or logical entities such as software components, database tables, etc. The *state* of the system is denoted by a vector $\mathbf{X} = (X_1, ..., X_n)$ of Boolean variables, where $X_i = 1$ denotes faulty state and $X_i = 0$ denotes OK state of node N_i. Lower-case letters denote the values of the corresponding variables, e.g. $\mathbf{x} = (x_1, ..., x_n)$ denotes a particular assignment of node values.

A *probe*, or *test* T is a method of obtaining information about the system components. The set of components tested by a probe T (i.e. the components T depends on) is denoted $N(T) \subseteq \{N_1, ..., N_n\}$. A probe either succeeds or fails: if it succeeds (denoted $T = 0$), then every component it tests is OK; it fails (denoted $T = 1$) if *any* of the components it tests is faulty.

We will consider the following two problems: *fault detection* problem is to discover if there is at least one faulty components in a system, while *fault diagnosis (fault localization)* problem is to find all faulty components in the system. Fault diagnosis will be also called "problem diagnosis" or "problem determination". Solving both problems requires: (a) selection of probes to run and (b) inference about the state of components given the outcomes of these probes.

It is useful to introduce the notion of a *dependency matrix* to capture the relationships between system states and probes. Given any set of nodes $\mathbf{N} = \{N_1, N_2, ..., N_n\}$ and probes, or tests $\mathbf{T} = \{T_1, T_2, ..., T_r\}$, the dependency matrix $D_{\mathbf{T},\mathbf{N}}$ is

given by:

$$\begin{aligned} D_{\mathbf{T},\mathbf{N}}(i,j) &= 1 \text{ if } N_j \in N(T_i) \\ &= 0 \text{ otherwise.} \end{aligned}$$

$D_{\mathbf{T},\mathbf{N}}$ is an r-by-n matrix, where each row represents a probe and each column represents a node.

It is also convenient to use the dependency matrix as an explicit representation of possible fault combinations that can occur in a system. In case of single fault only, we can view each column in the dependency matrix as a fault at a particular node, and only need to add a column representing no-fault situation. In general, multiple simultaneous failures can be handled by adding multiple columns to the dependency matrix, each additional column F_j representing a set of failed nodes. This representation will be called an *extended dependency matrix*. Note that each F_j corresponds to some state of the system, i.e. some vector $\mathbf{x} = (x_1, ..., x_n)$.

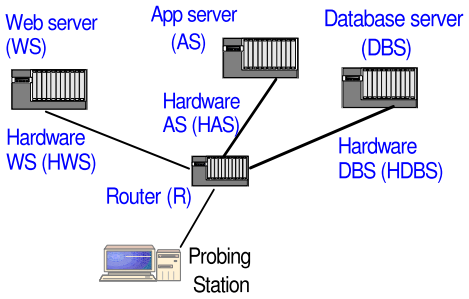
From now on, we will mostly use the extended dependency matrix, calling it simply the dependency matrix if no confusion arises; we will also use a similar notation $D_{\mathbf{T},\mathbf{F}}$ for extended dependency matrix, where $\mathbf{F}$ is a set of all possible fault combinations, or system states, corresponding to the columns of extended dependency matrix. The extended dependency matrix $D_{\mathbf{T},\mathbf{F}}$ is defined as follows:

$$\begin{aligned} D_{\mathbf{T},\mathbf{F}}(i,j) &= 1 \text{ if } F_j \bigcap N(T_i) \neq \phi \\ &= 0 \text{ otherwise.} \end{aligned}$$

$D_{\mathbf{T},\mathbf{F}}$ is an r-by-m matrix, where m is the number of possible fault combinations.

Unfortunately, the extended dependency matrix is intractable in general case since $m = 2^n$. Thus, for efficiency reasons, we will make assumptions about the maximum possible faults in the system, e.g. we will often assume single node failure. In this case there are only $n + 1$ possible states of the system (including no-fault situation), and thus only one column is added to the dependency matrix.

Example 1: Figure 2a shows an example of a simple benchmark system that contains a probing station, which sends various probes through a common router to a Web Server (WS), Application Server (AS), and Database Server (DBS). Note that WS, AS and DBS represent the OK/not OK state of the corresponding applications running on these machines, while HWS, HAS and $HDBS$ denote "hardware" problems with WS, AS, and DBS, respectively (in our case, HWS is OK if WS can be reached by $ping$ command; however, the web server application may not be running, and thus WS is not OK). Also, R will denote the state of the router, while NF corresponds to no-failure situation. The dependency matrix shown in Figure 2b includes the following probes:
"main" probe called pWS, attempts to open a web page on WS, which also runs an application on AS, which in its turn sends a query to a database on DBS. The outcome of this probe depends on the state (i.e., OK/not OK) of all components, i.e. WS, HWS, AS, HAS, DBS, and $HDBS$, as well as on the state of the router R. Thus, the row of the probe pWS contains ones in all columns (i.e., fails if any of these components fail).



(a)

Dependency matrix

Problem / Probe	WS	AS	DBS	R	HWS	HAS	HDBS	NF
pWS	1	1	1	1	1	1	1	0
pAS	0	1	1	1	0	1	1	0
pDBS	0	0	1	1	0	0	1	0
pingR	0	0	0	1	0	0	0	0
pingWS	0	0	0	1	1	0	0	0
pingAS	0	0	0	1	0	1	0	0
pingDBS	0	0	0	1	0	0	1	0

Probes:
pWS - Web Page access, **pAS**- Application Server access, **pDBS** - Database query, **pingR** – ping router, **pingWS** – ping Web Server, **pingAS** - ping Application server, **pingDBS** - ping Database Server

(b)

Fig. 2. (a) A simple benchmark distributed system with one probe station and 7 probes; (b) dependency matrix for the system in (a).

- probe pAS calls an application on AS which sends a query to the database on DBS; thus the probe depends on the states of AS, HAS, DBS, $HDBS$, R.
- probe $pDBS$ sends a query to the database on DBS, and thus depends on DBS, $HDBS$ and R.
- probes $pingR$, $pingWS$, $pingAS$ and $pingDBS$ are simply $ping$ commands to the router and the corresponding servers.

III. Overview of Our Approach

In this section, we introduce our adaptive real-time diagnosis approach which is outlined in Figure 3. Probe-stations issue the probes which traverse different parts of the network. The results of the probes are analyzed to infer what problems might be occurring. If additional information is needed in order to locate the problem, the most useful probes to send next are determined and sent. When additional probe results are received further inferences are made and the process repeats until the fault is localized. Implementing active probing requires developing solutions for the following issues:

1) Initial probe set selection for *fault detection*: a small subset of probes that "cover" all nodes must be pre-selected and run on a scheduled basis, so that when a fault occurs somewhere in the network we can immediately detect it.

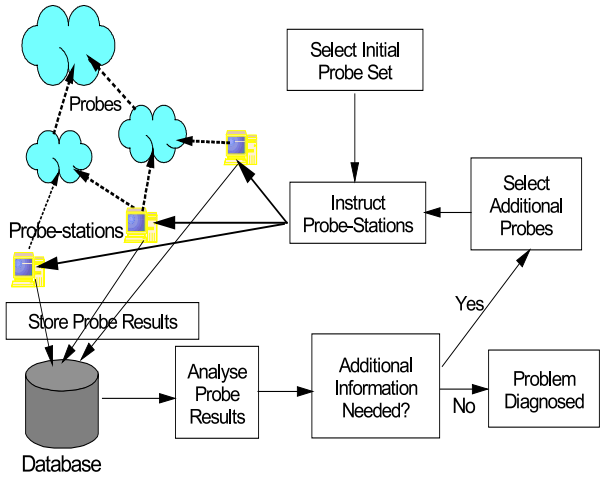


Fig. 3. An overview of adaptive real-time diagnosis system.

2) *Active, on-line probing* for *fault diagnosis*: the most-informative next probe must be selected and sent, based on the analysis of previous probe results. This approach extends an off-line, pre-planned probe selection for fault diagnosis.
3) *Analysis of probe results (inference)*: the probe results must be integrated and analyzed in order to diagnose the current faults in the systems.

Once a fault is detected, active probing and inference procedures will be performed repeatedly until the fault is uniquely diagnosed (or there are no more informative probes left).

In the following sections, we describe in more detail the three components of adaptive diagnosis mentioned above, i.e. (1) off-line probe selection for fault detection, (2) online (active) probing, and (3) probabilistic inference. For completeness sake, we also describe existing off-line, or pre-planned, probe selection approaches for diagnosis. We then quantify the advantages of an active probing methodology when compared with an entirely pre-planned approach, and present experimental results that show that active probing can greatly the number of probes and the time needed to perform fault diagnosis.

IV. Pre-Planned Probing

A. Probe Selection for Fault Detection

The task of *fault detection* is to find the smallest set of probes such that, no matter which problem occurs, there is some probe that will fail; i.e. will detect that a problem has occurred somewhere. Using the dependency matrix formulation, this corresponds to finding the smallest set of rows (a subset of all probes T) such that each column has a nonzero entry in one of these rows (these probes are said to "cover" all nodes). The corresponding decision problem is to determine, given the dependency matrix $D_{\mathbf{T},\mathbf{N}}$ and some positive integer $k \leq r$ (recall that r is the number of probes, i.e. rows in $D_{\mathbf{T},\mathbf{N}}$), whether T contains such a covering subset of size at most k. We call this decision problem Probe Set Selection for Fault Detection.

Probe-set selection for Fault Detection

Input: A set of available probes $\mathbf{T}$ and a prior distribution over system states $P(\mathbf{X})$.
Output: A set $\mathbf{S}$ of probes sufficient for fault detection.
Initialize: $\mathbf{S} = \emptyset$.
1. Simulate no-fault situation, i.e. assign $X_i = 0$ for all $1 \leq i \leq n$.
2. $\mathbf{T_a}$ = ActiveProbing($\mathbf{T}$,$P(\mathbf{X})$).
Return $\mathbf{S}$

Fig. 4. Algorithm for initial probe set selection for fault detection.

Proposition 1: Probe Set Selection for Fault Detection is NP-hard.

Proof: The problem is precisely the Minimum Set Cover problem, which is known to be NP-hard [11]. ■

Although finding minimal probe set for fault detection is NP-hard, there are simple and efficient heuristic search algorithms available. For example, as we show later, we can simply use the active probing approach described in this paper, applying it to the no-fault situation in the network (see Figure 4). We will see that in such case active probing keeps selecting probes until there is no uncertainty left about the state of the system (i.e. until the no-fault situation is confirmed), and produces a set of probes covering all nodes – exactly what we need for problem detection.

B. Probe Selection for Fault Diagnosis

The task of fault detection should be distinguished from the task of fault diagnosis, which requires not simply detecting that a problem has occurred, but also identifying, from the results of the probes, precisely which problem has occurred.

For completeness sake, we will now provide an overview of a pre-planned probe selection approach to fault diagnosis proposed earlier in [9], [10], since it will be used as a baseline for comparison with the active probing approach proposed in this paper. In addition, we will provide new complexity results, and generalize the previously proposed pre-planned greedy algorithm using information-theoretic framework, so that it can be easily modified into online, adaptive probe-selection.

In pre-planned probing, our objective is to select the smallest subset of probes that can diagnose the same set of possible faults (system states) as the original probe set. Let us consider the (extended dependency matrix $D_{\mathbf{T},\mathbf{F}}$ where each column in $\mathbf{F}$ represents a unique system state (combination of faults). Given a particular system state corresponding to column-vector F_j, it is easy to see that the vector of probe outcomes $\mathbf{t} = (t_1, ..., t_r) = (D_{1j}, ..., D_{rj})$: every probe T_i that contains nodes in F_j will fail, producing $T_i = 1 = D_{ij}$ in the column, and every probe that does not contain nodes in F_j must be OK ($T_i = 0 = D_{ij}$). Then the problem can be formulated as finding the smallest probe set such that every column of the dependency matrix is unique, since in that case exactly which state has occurred can be determined from the outcomes of all the probes.

We now formally define the associated decision problem and prove that it is NP-hard. Given a set of faults $\mathbf{F}$, a

set of probes $\mathbf{T}$, and a positive integer $k \leq r$, we want to determine whether $\mathbf{T}$ contains a subset $\mathbf{T}'$ of size at most k such that for every pair of distinct faults $f_1, f_2 \in F$, there is a probe $T \in \mathbf{T}'$ that intersects exactly one of f_1 and f_2 (thus T distinguishes between f_1 and f_2); or equivalently $\mathbf{T}'$ is such that the columns of the dependency matrix $D_{\mathbf{T}',\mathbf{F}}$ are all unique. We call this decision problem PROBE SET SELECTION FOR FAULT DIAGNOSIS.

Proposition 2: PROBE SET SELECTION FOR FAULT DIAGNOSIS is NP-hard.

Proof: PROBE SET SELECTION FOR FAULT DIAGNOSIS can be shown to be NP-hard via a reduction from 3-DIMENSIONAL MATCHING. This problem (see [12]) is however not very well-known, so it is instructive to reduce PROBE SET SELECTION FOR FAULT DIAGNOSIS from PROBE SET SELECTION FOR FAULT DETECTION. Intuitively, FAULT LOCALIZATION is a harder problem than FAULT DETECTION. However, because for any given instance the optimal solutions of these two problems can be very different, the proof is not straight-forward; the details can be found in the Appendix A. ■

Although the tasks of finding the smallest probe sets for fault diagnosis is NP-hard, there exist efficient polynomial-time approximation algorithms that perform well in practice. We now present an overview of two such algorithms – greedy search and subtractive search [9], [10]. We reformulate and generalize the greedy search algorithm proposed in [9], [10] using an information-theoretic framework.

Greedy search starts with the empty set and adds at each step the "best" of the remaining probes. The "best" probe is the one which maximizes the information gained about the system state, in the sense defined precisely below. *Subtractive search* starts with the complete set of available probes, considers each one in turn, and discards it if it is not needed. Neither algorithm is optimal in general - experimental results comparing their performance with the true minimum probe set size are given in Section VIII-B.1.

The *greedy search* approach chooses the next probe by maximizing the information gained about the system state $\mathbf{X}$, given the previous probes. Formally, let us assume some prior probability distribution $P(\mathbf{X})$ over possible system states. We are looking for a probe

$$Y^* = \arg \max_{Y \in \mathbf{T} \setminus \mathbf{T}'} I(\mathbf{X}; Y | \mathbf{T}'), \tag{1}$$

where $I(\mathbf{X}; Y|\mathbf{T}')$ is the conditional mutual information of $\mathbf{X}$ and probe Y, given the previously selected probes $\mathbf{T}'$. Since $I(\mathbf{X}; Y|\mathbf{T}') = H(\mathbf{X}|\mathbf{T}') - H(\mathbf{X}|Y, \mathbf{T}')$, where $H(\mathbf{X}|Y)$ is the conditional entropy of $\mathbf{X}$ given Y (see [13]), the most-informative test Y^* minimizes the conditional entropy of $\mathbf{X}$, i.e. the amount of uncertainty about the system state.

The algorithm is shown in Figure 5a. If the initial set of available probes $\mathbf{T}$ is of size r, $O(r^2)$ conditional entropy calculations are required, since at each step the information gain obtained by each of the remaining probes must be computed. Note that

$$H(\mathbf{X}|Y, \mathbf{T}) = -\sum_{\mathbf{x}} \sum_{y_j} \sum_{\mathbf{t}} P(\mathbf{x}, y, \mathbf{t}) \log P(\mathbf{x}|y, \mathbf{t}) \tag{2}$$

Probe-Set Selection: Greedy Search

Input: A set of available probes $\mathbf{T}$ and a prior distribution over system states $P(\mathbf{X})$.
Output: A subset $\mathbf{T}' \subseteq \mathbf{T}$ of probes.
Initialize: $\mathbf{T}' = \emptyset$
do1. select most-informative next probe:
$Y^* = \arg \max_{Y \in \mathbf{T} \setminus \mathbf{T}'} I(\mathbf{X}; Y|\mathbf{T}')$
2. update probe set: $\mathbf{T}' = \mathbf{T}' \cup \{Y\}$
while $\exists Y \in \mathbf{T} \setminus \mathbf{T}'$ such that $I(\mathbf{X}; Y|\mathbf{T}') > 0$
Return T'.

(a)

Probe-Set Selection: Subtractive Search

Input: A set of available probes $\mathbf{T}$ and a prior distribution over system states $P(\mathbf{X})$.
Output: A subset $\mathbf{T}' \subseteq \mathbf{T}$ of probes.
Initialize: $\mathbf{T}' = \mathbf{T} = \{T_1, T_2, ..., T_r\}$.
for $i = 1$ to r
remove probe if it is not needed, i.e.
if $I(\mathbf{X}; p_i|\mathbf{T}' \setminus \{T_i\}) = 0$,
then $\mathbf{T}' = \mathbf{T}' \setminus \{T_i\}$
Return $\mathbf{T}'$.

(b)

Fig. 5. (a) Greedy Search for Probe-Set Selection. (b) Subtractive Search for Probe-Set Selection.

so computing the information gain can be quite costly in the general case, as it requires summation over all non-zero-probability states and outcomes of the current probe set and the next probe.

The version of greedy search proposed in [9], [10] used the simplifying assumptions of only one component failing at a time, thus yielding only $n+1$ system states (including the case of no failure). Also, all states were assumed to have equal prior probabilities. These assumptions considerably simplified the computation. Let us consider an extended dependency matrix of size $r \times (n+1)$. A probe set cannot distinguish between system states whose columns in the dependency matrix are identical. Since this is an equivalence relation between states (columns), it induces a decomposition of the states into an exhaustive collection of disjoint subsets, and it is easy to show that:

$$H(\mathbf{X}|Y, \mathbf{T}) = \sum_{i=1}^{k} \frac{n_i}{m} \log n_i$$

where $m = n + 1$ is the total number of states and n_i is the number of states (columns) in the i'th subset of the decomposition induced by $\mathbf{T}$.

This expression has a natural interpretation. Since there are n_i states in the i'th subset and each probe has two possible outcomes, at least $\log n_i$ additional probes are needed to further decompose the i'th subset into singletons, thereby enabling any single node failure to be diagnosed. Since the true failure lies in the i'th subset with probability n_i/m, the

conditional entropy is simply the expected minimal number of additional probes needed to localize the failure.

The greedy algorithm can of course be generalized by adding the best subset of t of the remaining probes at each step, requiring $O(r^{t+1})$ conditional entropy calculations.

The *subtractive search* algorithm starts with the complete set of available probes, considers each one in turn, and discards it if it is not needed, i.e. if removing it does not result in any loss of information about the system state. The algorithm is shown in Figure 5b. Each probe is considered only once, so $O(r)$ conditional entropy computations are required.

Once a probe set is selected, it can be scheduled to run through the system, and an online diagnosis can simply match the vector of probe outcomes to columns in the dependency matrix. Each system state, i.e. the (extended) dependency matrix column, will have a unique "signature" vector of probe outcomes.

V. Active Probing

In this section, we propose a novel, more efficient approach to probe-based diagnosis called **active probing**. The selection of probes is now more adaptive to the current system state as it depends on the results of earlier probes (is "context-sensitive"). The advantage of this approach is that fewer probes can be used on average than if the entire probe set has to be pre-planned. However additional inferential machinery is required. We describe an algorithm for active probing and then present experimental results that show that active probing can greatly reduce the number of probes needed to perform fault localization.

An active probing algorithm is described in Figure 6a. It takes as input a set of probes $\mathbf{T}$ available for selection, and a prior distribution $P(\mathbf{X})$ over system states. The algorithm maintains the current *belief* about the system state, $Belief(\mathbf{X}) = P(\mathbf{X}|\mathbf{T_a})$ where $\mathbf{T_a}$ is the current set of probes and their outcomes. The prior distribution is used to initialize $Belief(\mathbf{X})$. Similarly to greedy search, the active probing approach is always looking for the next probe Y^* that maximizes information gained about the system state $\mathbf{X}$ (see step 1 in both algorithms). However, there is an important difference: active probing has an advantage of *knowing* the previous probe outcomes; thus, at iteration k it conditions on probe outcomes $\mathbf{T_a} = \{Y_1^* = y_1^*, ..., Y_{k-1}^* = y_{k-1}^*, \}$, while the pre-planned greedy approach must average over all possible outcomes of previously selected probes $\mathbf{T}' = \{Y_1^*, ..., Y_{k-1}^*\}$. Namely, active probing in step 1 finds Y that maximizes $I(\mathbf{X}; Y|\mathbf{T_a})$, i.e. minimizes

$$H(\mathbf{X}|Y, \mathbf{T_a}) = -\sum_{\mathbf{x}} \sum_{y_j} P(\mathbf{x}, y|\mathbf{T_a}) \log P(\mathbf{x}|y, \mathbf{T_a}), \quad (3)$$

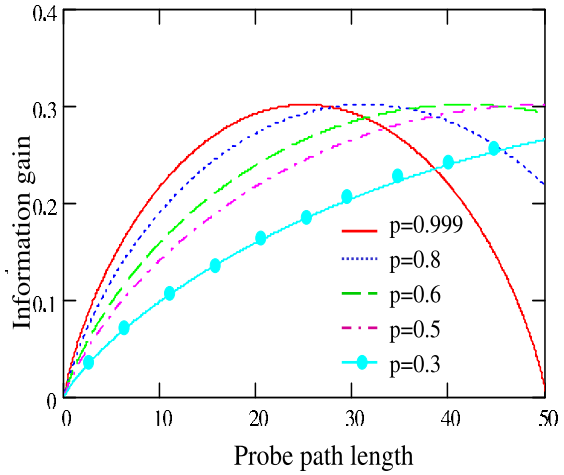
which is much faster (exponentially in the number of previous probes) than minimization performed by greedy search in the equation 2 which has an additional sum over all possible outcomes of previously selected probes. However, as we will demonstrate empirically, the main advantage of active probing besides faster probe selection is a much smaller, on average, number of probes needed for diagnosis, and thus much faster time to diagnose a problem.

Active Probing

Input: A set of available probes $\mathbf{T}$ and a prior distribution over system states $P(\mathbf{X})$.
Output: A set $\mathbf{T}_a$ of probes and their outcomes, posterior distribution $Belief(\mathbf{X})$ and its support $\mathbf{S}$.
Initialize: $Belief(\mathbf{X}) = P(\mathbf{X})$, $\mathbf{T_a} = \emptyset$, $\mathbf{S}$ = support of $P(\mathbf{X})$.
do
1. select current most-informative probe: $Y^* = \arg\max_{Y \in \mathbf{T} \setminus \mathbf{T_a}} I(\mathbf{X}; Y|\mathbf{T_a})$
2. execute Y^*; it returns $Y^* = y^*$ (0 or 1)
3. update $\mathbf{T}_a = \mathbf{T}_a \cup \{Y^* = y^*\}$
4. update $Belief(\mathbf{X}) = P(\mathbf{X}|\mathbf{T_a})$

while $\exists Y \in T$ such that $I(\mathbf{X}; Y|\mathbf{T}_a) > 0$
Return $\mathbf{T_a}$, $Belief(\mathbf{X})$, S =support of $Belief(\mathbf{X})$.

(a)



(b)

Fig. 6. (a) Active Probing algorithm for probabilistic diagnosis with most-informative probe selection; (b) Information gain as a function of probe's 'effective' length (the number of nodes on the probe's path having non-zero fault probability) for various probabilities p of a single fault in any component (assuming n=50 components).

The active-probing diagnosis algorithm works as follows. It selects the next probe to run (step 1) and waits for the results (step 2), then it updates both the set of active probes executed (step 3) and the current belief about the system's state (step 4). Belief updating, or *probabilistic inference*, will be considered in more details in the following section. Steps 1-4 are repeated until no more information can be obtained about the system state, and thus the diagnosis cannot be improved. The algorithm outputs a set of active probes $\mathbf{T_a}$ that were actually used during diagnosis (which often turns out to be significantly smaller than the original set $\mathbf{T}$), the posterior distribution over components after receiving the probe outcomes, $P(\mathbf{X}|\mathbf{T_a})$, and the *support* of the distribution (the components with non-zero-probability). Let us also define the *effective* length of a probe as the number of nodes on probe's path that currently have non-zero fault probability (i.e., belong to the support of the current $Belief(\mathbf{X})$).

Active Probing for fault detection. Note that in case of no-failure situation, the active probing algorithm will always

produce a set of probes that cover all nodes (assuming the initial probe set allows to cover all nodes). Indeed, its stopping condition implies there is no probe that can have a non-zero information about the system state, $I(\mathbf{X}; Y|\mathbf{T}_a)$. It is easy to see that while there is at least a single node not covered by a probe, the system state is still uncertain, and this uncertainty can be decreased by running an additional probe that goes through this node. Once all nodes are covered, the conditional entropy of the system $I(\mathbf{X}; T_a)$ becomes zero, and the active probing stops. Thus, simulating no-fault situation and running active probing will produce a set of probes sufficient for problem detection.

VI. Analysis of Probe Results: Probabilistic Inference

A. Diagnosis under Simplifying Assumptions

In our current implementation of active probing, we made simplifying assumptions of having no more than s simultaneous faults (usually, setting $s = 1$ as having simultaneous faults was highly unlikely in systems we considered), which yields more than 2^s system states. For small s the explicit state space representation is relatively small (e.g., it is linear in the number of nodes for single-fault assumption), and thus we can easily update the joint distribution $P(\mathbf{X})$ over all possible states of the system. However, for generic multi-fault diagnosis, more efficient algorithms, including approximations, are needed. An extension to generic multi-fault diagnosis will be considered in the subsequent sections.

Besides single-fault assumption, we also used a simple expressions for priors, assuming $1 - p$ probability of no-fault situation, and uniformly distributed probability mass p among the remaining states (possible single component failures), which gave us $P(X_i = 1, X_j = 0 \forall j \neq i) = p/n$. In this case the information gain of a probe can be computed quite efficiently (see Appendix B for details):

$$I(X;T) = I(p,n,k) = -(1-p)\log(1-p) - p\log\frac{p}{n}$$
$$-k\frac{p}{n}\log k + (1-p)\log\frac{1-p}{1-k\frac{p}{n}} + (n-k)\frac{p}{n}\log\frac{p}{n-pk}. \quad (4)$$

Figure 6b plots the information gain of a probe as a function of its effective length k, for $n = 50$ nodes and for various fault priors. It is more beneficial to send probes with larger effective length if the probability of fault p is small. However, once a fault is detected ($p = 1$), the most informative probe (i.e. a probe attaining the maximum information gain) is one whose effective length is closest to half the number of nodes that are possibly faulty.

In the single-fault case, if the complete set of available probes is sufficient for diagnosis, the support set S output by the algorithm will contain the unique component that is down. In general, the active probing algorithm can be applied to any multiple-fault situation; however, its complexity increases with an increasing number of simultaneous faults and depends on the efficiency of representing the joint probability $P(\mathbf{X})$ and the efficiency of probabilistic inference and information-gain computation required for active probe selection. It may also become impossible to update the joint distribution, i.e. $Belief(\mathbf{X})$. Instead, we will have to modify the line 4 in the Active Probing algorithm: given the previous probe observations, instead of updating joint belief, we may need to update the beliefs of individual nodes and return their most-likely values, or to find directly the most-likely state of the system as a 0/1 assignment to all nodes. The problem of multi-fault diagnosis is considered in the following section.

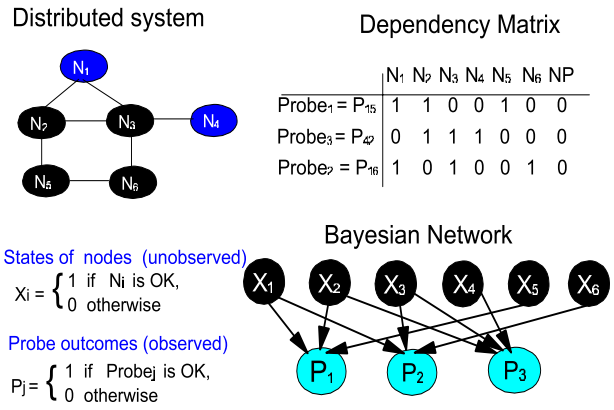


Fig. 7. A mapping from dependency matrix to a Bayesian network.

B. Multi-Fault Diagnosis using Bayesian Networks

Active probing algorithm described in the previous section must use some representation of the joint probability distribution $P(\mathbf{X})$ and a belief updating method. While in case of small number of possible states (e.g., $n + 1$ states under single-fault assumption) the joint distribution $P(\mathbf{X})$ can be specified explicitly, this representation becomes intractable in the general case of multiple faults. In this case, fault diagnosis algorithms can benefit from using a probabilistic graphical model such as Bayesian network [14], which describes the probability distribution over the system states in a compact form. For example, the (original) dependency matrix can be easily mapped to a two-layer Bayesian network where each component X_i corresponds to an upper-level variable representing the state of node N_i, each probes T_j corresponds to a lower-layer variable, and each subset $N(T_j)$ corresponds to the set of *parents* of variable T_j (i.e., nodes pointing to T_j) in the Bayesian network, also denoted $\mathbf{pa_j}$ (see Figure 7). Recall that $X_i = 1$ and $T_j = 1$ if node N_i and probe T_j are faulty, respectively; and $X_i = 0$ and $T_j = 0$ otherwise. Then the joint probability distribution can be written as

$$P(\mathbf{x}, \mathbf{t}) = \prod_{i=1}^{n} P(x_i) \prod_{j=1}^{m} P(t_j|\mathbf{pa_j}),$$

assuming that the state variables X_i's are marginally independent, and that each probe outcome depends only on the components tested by this probe. $P(x_i)$ specifies the prior probabilities of the system states, while the *conditional probabilities distributions (CPDs)* $P(t_j|\mathbf{pa_j})$ describe the dependency of probe outcomes on the components tested. In the absence of noise in the probe outcomes, all CPDs are deterministic functions. Since a probe succeeds if and only if

all its components are OK, a probe outcome is a logical-OR function of its components, i.e. $T_i = X_{i_1} \vee ... \vee X_{i_k}$, where $\vee$ denotes logical OR, and $X_{i_1}, ..., X_{i_k}$ are all the nodes probe T_i goes through.

In practice, however, this relationship may be disturbed by noise in the measurements. For example, a probe can fail even though all the nodes it goes through are OK (e.g., due to packet loss). Conversely, there is a chance that a probe succeeds even if a node on its path has failed (e.g., dynamic routing may result in the probe following a different path). Such uncertainties yield a so-called *noisy-OR*, where every probe T_j and component X_i on probe's path are associated with a *noise* parameter q_{ij}, also called *inhibition probability*, or *link probability* – a small probability that probe T_j succeeds even if node X_i on its path has failed. There is also a parameter called the *leak probability* which accounts for the cases of a probe failing even when all the nodes on its path are OK. Finally, noisy-OR model assumes *causal independence*, i.e. it assumes that different causes (e.g., node failures) contribute independently to a common effect (probe failure). The the conditional probability distribution for each probe T_j can be written as

$$P(t_j = 0 | x_1, \ldots, x_k) = (1-l) \prod_{x_i=1}^{k} q_{ij} = (1-l) \prod_{i=1}^{n} q_{ij}^{x_i}, \quad (5)$$

(6)

where $X_1, \ldots, X_k$ is the set of parents $\mathbf{pa_j}$ of T_j.

Once a Bayesian network is specified, we can use *probabilistic inference* for diagnosis. The most common probabilistic inference task is *belief updating*, i.e. finding $P(\mathbf{Z}|\mathbf{Y} = \mathbf{y})$, the posterior probability of set of variables $\mathbf{Z}$ given observations of some other variables $P(\mathbf{Y} = \mathbf{y})$. For example, we may want to find $P(X_i|\mathbf{T} = \mathbf{t})$, the posterior probability of fault in component X_i given the probe outcomes. Then we can return as a result of diagnosis the most likely values of each node X_i, namely, a vector $\mathbf{x}' = (x'_1, ..., x'_n)$ where $x'_i = \arg\max_{x_i} P(x_i|\mathbf{t})$, $i = 1, ..., n$.

Alternative approach to diagnosis is to find the most-likely state called the *maximum probable explanation (MPE)*, i.e. a most-likely assignment to all X_i nodes given the probe outcomes, i.e. $\mathbf{x}^* = \arg\max_{\mathbf{x}} P(\mathbf{x}|\mathbf{t})$. Since $P(\mathbf{x}|\mathbf{t}) = \frac{P(\mathbf{x},\mathbf{t})}{P(\mathbf{t})}$, where $P(\mathbf{t})$ does not depend on $\mathbf{x}$, we get $\mathbf{x}^* = \arg\max_{\mathbf{x}} P(\mathbf{x}, \mathbf{t})$.

Both belief updating and finding MPE are known to be NP-hard problems [15]; particularly, the complexity of commonly used inference algorithms such as *join-tree* algorithm [14], [16], and closely related *variable-elimination* techniques [17], [18], is known to be $O(n \cdot exp(w^*))$ where n is the number of unobserved nodes in the network and w^* is the size of largest probabilistic dependency recorded during inference, which corresponds to largest clique induced in the network and is called *induced width*, or *treewidth* [18]. Typical approaches to handling this complexity are exploiting the problem structure (when the induced width is limited) or using approximate techniques.

A popular approximation approach is to restrict the complexity by focusing only on local interactions. In our previous work on probing [10] we investigated the *mini-bucket* approximation algorithms [19], [20] that bound the size of functions created by variable-elimination inference. The algorithms performed nearly optimal when the level of noise (link probabilities) was small. In this paper, we provide a theoretical analysis of the results presented in [10] for the simplest member of the mini-bucket algorithmic family called *greedy-mpe*; we will derive a necessary condition for the algorithm to be exact which explains our earlier empirical results [10].

Note that the mini-bucket scheme is closely related to other local approximations, such as *iterative belief propagation (IBP)* and *generalized belief propagation (GBP)* algorithms [21], that recently became the state-of-the-art approximation techniques successfully used in many practical applications of Bayesian networks. The mini-bucket algorithms can be viewed as simplified, non-iterative versions of those approaches, which nevertheless performs surprisingly well in low-noise problems. In order to gain a better theoretical understanding of this success we decided to focus first on a simple scheme rather on more advanced techniques.

In the next sections, we describe the diagnosis task as MPE problem and algorithms in more detail, and provide theoretical analysis.

C. Diagnosis complexity and approximations

In this section, we focus on the formulation of diagnosis problem as finding the most-probable explanation, or MPE. We focus first on the MPE diagnosis in the absence of noise (i.e., for deterministic test outcomes). The deterministic CPDs reduce to a set of constraints imposed by the test outcomes on the values of $X_1, ..., X_n$. For example, in the fault diagnosis domain, each probe outcome $T_i = t_i$ imposes a logical-OR constraint $t_i = x_{i_1} \vee ... \vee x_{i_k}$ on the values of its parent nodes $X_{i_1}, ..., X_{i_k}$. The MPE diagnosis becomes a constrained optimization problem of finding $\mathbf{x}^* = \arg\max_{x_1,\ldots,x_n} \prod_{j=1}^{n} P(x_j)$ subject to those constraints. In a particular case of uniform priors $P(x_j)$, diagnosis is reduced to solving a constraint satisfaction problem. The problem can also be cast as a constraint satisfaction rather than optimization if there exist a unique solution satisfying the constraints (see [9] for more details on how to construct such probe sets).

Although constrained optimization and constraint satisfaction problems (CSPs) are generally NP-hard, it is interesting to note that the probing domain yields a tractable set of constraints.

Proposition 2: A set of *disjunctive clauses* $t_j = x_{j_1} \vee ... \vee x_{j_k}$, $j = 1, ..., m$ over a set of variables $X_1, ..., X_n$, where $x_i \in \{0, 1\}, t_j \in \{0, 1\}$ for $i = 1, ..., n$ and $j = 1, ..., m$, can be solved in $O(n)$ time.

Indeed, each successful probe yields a constraint $x_{i_1} \vee ... \vee x_{i_k} = 0$ which implies that all nodes X_i on its path are OK ($x_i = 0$); the rest of the nodes are only included in constraints $x_{i_1} \vee ... \vee x_{i_k} = 1$ imposed by failed probes. A simple $O(n)$-time algorithm would find a satisfying assignment to all nodes by assigning 0 to every node appearing on the path of a

successful probe, and 1 to the rest of nodes[2].

In the presence of noise, the MPE diagnosis task can be written as finding $\mathbf{x}^* = \arg\max_{x_1} \ldots \max_{x_n} \prod_i P(x_i|pa_i) =$

$$= \arg\max_{x_1} F_1(x_1) \ldots \max_{x_n} F_n(x_n, S_n), \quad (7)$$

where each $F_i(x_i, \mathbf{S_i}) = \prod_{\mathbf{x_k}} \mathbf{P}(\mathbf{x_k}|\mathbf{pa}(\mathbf{x_k}))$ is the product of all probabilistic components involving X_i and a subset of lower-index variables $\mathbf{S_i} \subseteq \{\mathbf{X_1}, ..., \mathbf{X_{i-1}}\}$, but *not* involving any X_j for $j > i$. The set of all such components is also called the *bucket* of X_i [18]. An exact algorithm for finding MPE solution, called *elim-mpe* [18], uses *variable-elimination* (also called *bucket-elimination*) as a preprocessing: it computes the product of functions in the bucket of each variable X_i, from $i = n$ to $i = 1$ (i.e., from right to left in the equation 7), maximizes it over X_i, and *propagates* the resulting function $f(\cdot)$ to the bucket of its highest-order variable. Once variable-elimination is completed, the algorithm finds an optimal solution by a backtrack-free greedy procedure that, going from $i = 1$ to $i = n$ (i.e., in the opposite direction to elimination), assigns $X_i = \arg\max_{x_i} F_i(x_i, \mathbf{S_i} = \mathbf{s_i})$ where $\mathbf{S_i} = \mathbf{s_i}$ is the current assignment to $\mathbf{S_i}$. It is shown that *elim-mpe* is guaranteed to find an optimal solution and that the complexity of the variable-elimination step is $O(n \cdot exp(w^*))$ where w^*, called the *induced width*, is the largest number of arguments among the functions (old and newly recorded) in all buckets [18]. For the probing domain, it is easy to show that $w^* \geq k$ where k is the maximum number parents of a probe node, and $w^* = n$ in the worst case.

Since the exact MPE diagnosis is intractable for large-scale networks, we focused on *local* approximation techniques. Particularly, we used a simple (O(n) time) backtrack-free greedy algorithm, called here *greedy-mpe*, which performs no variable-elimination preprocessing, and the simplest and fastest member of the *mini-bucket* approximation family, algorithm *approx-mpe(1)* [19], [20], that performs a very limited preprocessing similar to *relational arc-consistency* [20] in constraint networks.

The greedy algorithm *greedy-mpe* does no preprocessing (except for replacing observed variables with their values in all related function prior to algorithm's execution). It computes a suboptimal solution

$$\mathbf{x}' = (\arg\max_{x_1} F_1(x_1), \ldots, ..., \arg\max_{x_n} F_n(x_n, S_n = \mathbf{s_n})), \quad (8)$$

where $S_i = \mathbf{s_i}$, as before, denotes the current assignment to the variables in S_j computed during the previous $i - 1$ maximization steps.

Generally, the mini-bucket algorithms *approx-mpe(i)* perform a limited level of variable-elimination, similar to enforcing directional i-consistency, prior to the greedy assignment. The preprocessing allows to find an upper bound U on $M = \max_{\mathbf{x}} P(\mathbf{x}, \mathbf{t})$, where $\mathbf{t}$ is the evidence (clearly, $MPE = M/P(\mathbf{t})$), while the probability $L = P(\mathbf{x}', \mathbf{e})$ of their suboptimal solution provides an lower bound on M. Generally, L increases with the level of preprocessing controlled by i, thus allowing a flexible accuracy vs. efficiency trade-off. The algorithm returns the suboptimal solution $\mathbf{x}'$ and the upper and lower bounds, U and L, on M; ratio U/L is a measure of the approximation error.

In [10], the algorithms *greedy-mpe* and *approx-mpe(1)* were tested on the networks constructed in a way that guarantees the unique diagnosis in the absence of noise. Particularly, besides m tests each having r randomly selected parents, we also generated n *direct* tests $\hat{T}_i$, $i = 1, ..., n$, each having exactly one parent node X_i. It is easy to see that, for such networks, both *greedy-mpe* and *approx-mpe(1)* find an exact diagnosis in the absence of noise: *approx-mpe(1)* reduces to the unit-propagation, an equivalent of relational-arc-consistency, while *greedy-mpe*, applied along a *topological* order of variables in the network's directed acyclic graph (DAG)[3], immediately finds the correct assignment which simply equals the outcomes of the direct tests.

We then added noise in a form of non-zero link probability q (assumed to be equal for all nodes), and zero leak probability. The Figure 8 summarizes the results for 50 randomly generated networks with $n = 15$ unobserved nodes (having uniform fault priors $p = P(x_i = 0) = 0.5$), $n = 15$ direct probes, one for each node, and $m = 15$ noisy-OR probes, each with $r = 4$ randomly selected parents among the unobserved nodes, zero leak $l = 0$ probability. The link probability (noise level) q varied from 0.01 to 0.64, taking 15 different values; the results are shown for all noise levels together. For each network, 100 instances of evidence (probe outcomes) were generated by Monte-Carlo simulation of $\mathbf{x}$ and $\mathbf{t}$ according to their conditional distributions. Thus, we get 50x100=5000 samples for each value of noise q.

As demonstrated in Figure 8, the approximation accuracy of *greedy-mpe*, measured as L/M where $L = P(\mathbf{x}', \mathbf{t})$ and $M = P(\mathbf{x}^*, \mathbf{t})$, clearly increases with increasing value M, and therefore with the probability of the exact diagnosis, which also depends on the "diagnostic ability" of a probe set (for same probe set size, a better probe set yields a higher MPE diagnosis, and therefore, a better approximation quality). There is an interesting threshold phenomenon, observed both for *greedy-mpe* and for *approx-mpe(1)* solutions (the results for *approx-mpe(1)* are omitted due to space restrictions), and for various problem sizes n: the suboptimal solution $\mathbf{x}'$ found by algorithm *greedy-mpe* suddenly becomes (almost always) an exact solution $\mathbf{x}^*$ (i.e., $L/M = 1$, where $L = P(\mathbf{x}', \mathbf{t})$ and $M = P(\mathbf{x}^*, \mathbf{t})$) when $M > \theta$ where θ is some threshold value. For $n = 15$, the threshold is observed between $2e-6$ and $3e-6$. A theoretical analysis in the next section yields a quite accurate prediction of $\theta \approx 2.46e-6$.

1) Theoretical analysis: noise and approximation accuracy: We now provide a theoretical explanation for threshold behavior of approximation error reported in [10], at least for the simplest approximation algorithm *greedy-mpe*.

Let $BN = (G, P)$ be a Bayesian network, where $\mathbf{T} = \mathbf{t}$ is evidence, i.e. a value assignment $\mathbf{t}$ to a subset of variables

[2]Note that this is equivalent to applying *unit propagation* to a Horn theory – a propositional theory that contains a collection of disjunctive clauses, or disjuncts, where each disjunct includes no more than one positive literal.

[3]A *topological* (or ancestral) ordering of a DAG is an ordering where a child node never appears before its parent.

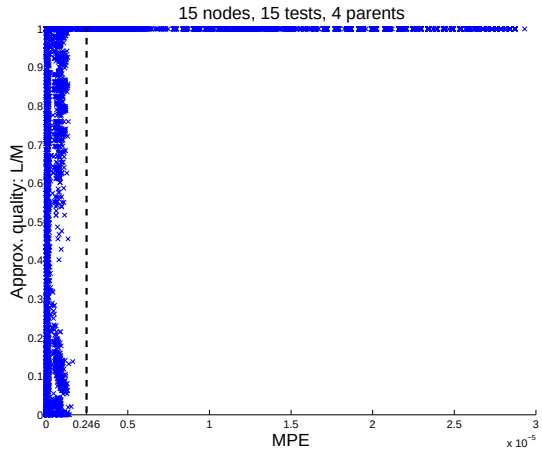


Fig. 8. The accuracy of the solution $\mathbf{x}'$ found by algorithm *greedy-mpe*, measured by L/M, where $L = P(\mathbf{x}', \mathbf{t})$ and $M = P(\mathbf{x}^*, \mathbf{t})$, versus M. The results obtained for *approx-mpe(1)* were quite similar to those for *greedy-mpe*.

$\mathbf{T} \subset \mathbf{X}$. We will also make an important assumption that the all observed variables are replaced by their values in all CPD functions. Also, recall that $F_i(x_i, \mathbf{s_i})$ is the product of functions in the bucket of X_i along the ordering o, given the assignment $\mathbf{s_i}$ of some variables in the previous buckets. Then

*Lemma 3 (*greedy-mpe *optimality):* Given a Bayesian network $BN = (G, P)$, an evidence assignment $\mathbf{T} = \mathbf{t}$ applied to all relevant probability functions, and a topological ordering o of unobserved nodes in the graph G, the algorithm *greedy-mpe* applied along o is guaranteed to find an optimal MPE solution if $P(\mathbf{x}', \mathbf{t}) \geq F_i(x_i, \mathbf{s_i'})$ for every $i = 1, ..., n$ and for every $x_i \neq x_i'$, where $\mathbf{S_i} = \mathbf{s_i'}$ is a partial assignment already found by *greedy-mpe*.

Proof: Clearly, the solution $\mathbf{x}'$ found by *greedy-mpe* is optimal, i.e. $\mathbf{x}' = \mathbf{x}^* = \arg\max_{\mathbf{x}} P(\mathbf{x}, \mathbf{t})$ if $P(\mathbf{x}', \mathbf{t}) \geq P(\mathbf{x}, \mathbf{t})$ for every $\mathbf{x} \neq \mathbf{x}'$. Since $\mathbf{x} \neq \mathbf{x}'$ implies $x_i \neq x_i'$ for some i (let us choose the smallest of such i's), by the condition of lemma we get $P(\mathbf{x}', \mathbf{t}) \geq F_i(x_i, \mathbf{s_i'})$, and, therefore, $P(\mathbf{x}', \mathbf{t}) \geq \prod_{j=1}^{n} F_j(x_j, \mathbf{s_j})$ since each $F_j(x_j, s_j)$ is a product of probabilities, and therefore, $0 \geq F_j(x_j, s_j) \geq 1$. But $\prod_{j=1}^{n} F_j(x_j, \mathbf{s_j}) = \mathbf{P}(\mathbf{x}, \mathbf{t})$ by equation 7, which concludes the proof. ■

We now discuss some particular classes of Bayesian networks that satisfy the conditions of lemma 3.

Lemma 4 (nearly-deterministic CPDs, no observations): Given a Bayesian network $BN = (G, P)$ having no observed variables, and all conditional (and prior) probabilities being nearly-deterministic, i.e. satisfying the condition $\max_{x_i} P(x_i|pa(X_i)) > 1 - \delta$, where $0 \leq \delta \leq 0.5$, algorithm *greedy-mpe* applied along a topological ordering o of G is guaranteed to find an optimal MPE assignment if $(1-\delta)^n \geq \delta$.

Proof: Given a topological ordering and no evidence variables, the bucket of every node X_i contains a single function $P(x_i|pa(X_i))$. Thus, the greedy solution $\mathbf{x}'$ yields $P(\mathbf{x}') = \prod_{i=1}^{n} \max_{x_i} P(x_i|pa(X_i)) = (1 - \delta)^n$, while any other $\mathbf{x}$ has the probability $P(\mathbf{x}) = \prod_{i=1}^{n} P(x_i|pa(X_i)) < \delta$ since for the very first i such that $x_i \neq x_i'$ we get $P(x_i|pa(X_i)) < \delta$ and this value can only decrease when multiplied by other probabilities $0 \leq P(x_j|pa(X_j)) \leq 1$. ■

Let us consider a simulation that happened to select only the most-likely values for $\hat{T}_i$ and T_i, i.e. $t_i' = \arg\max_{t_i} P(t_i|pa(T_i))$, which can be viewed as an error-free "transmission over a noisy channel". From 6 we get $\max_{t_i} P(t_i|pa(T_i)) \geq (1 - q)$; also, for any $t_i'' \neq \arg\max_{t_i} P(t_i|pa(T_i))$, $P(t_i''|pa(T_i)) < q$. It is easy to show (similarly to lemma 3) that algorithm *greedy-mpe* will find an assignment that produced this most-likely evidence, thus yielding $P(\mathbf{x}', \hat{\mathbf{t}}, \mathbf{t}) = \prod_{i=1}^{n} P(x_i) \prod_{i=1}^{n} P(\hat{t}_i) \prod_{i=1}^{n} P(t_i) > \frac{1}{2^n}(1-q)^{n+m}$. On the other hand, for any other $\mathbf{x}$ there exists $T_j = t_j$ where t_j is not the most-likely choice for T_j given $\mathbf{x}$, and thus $P(t_i|pa(t_i)) < q$ as can be seen from the noisy-OR definition. Thus, the greedy solution $\mathbf{x}'$ is guaranteed to be optimal once for any $\mathbf{x} \neq \mathbf{x}'$, $P(\mathbf{x}', \hat{\mathbf{t}}, \mathbf{t}) > P(\mathbf{x}, \hat{\mathbf{t}}, \mathbf{t})$, i.e. once $(1 - q)^{n+m} > q$ (the constant $\frac{1}{2^n}$ on both sides of the inequality was cancelled). Note that simulating an unlikely evidence yields a low joint probability $M = P(\mathbf{x}^*, \hat{\mathbf{t}}, \mathbf{t}) < q$ for the optimal diagnosis $\mathbf{x}^*$.

In our experiments, $n = m = 15$, thus resolving $(1-q)^{30} = q$ gives a threshold value $q \approx 0.0806$, and therefore $M = P(\mathbf{x}', \hat{\mathbf{t}}, \mathbf{t}) = \frac{1}{2^{15}}(1 - q)^{30} > \frac{1}{2^{15}} q \approx 2.46e - 6$, which is surprisingly close to the empirical threshold observed in Figure 8 which separates suboptimal from the optimal behavior of algorithm *greedy-mpe*.

VII. Empirical Results

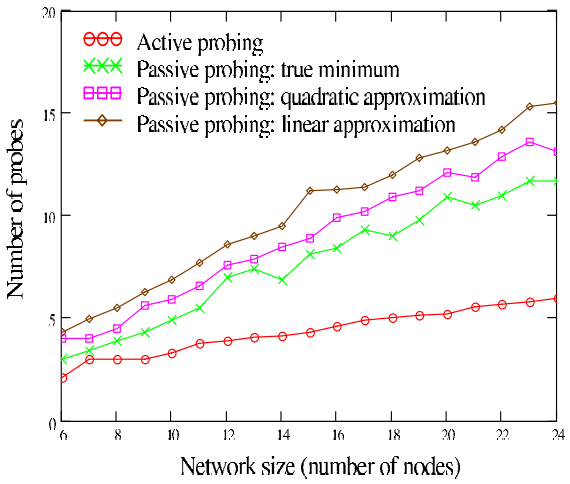
This section examines the empirical behavior of both pre-planned and active probing. For pre-planned probing the approximation algorithms find a probe set which is very close to the true minimum set size, and can be effectively used on large networks where finding the true minimum by exhaustive search is impractical. Active probing greatly reduces the number of probes needed, although at the expense of a more complex interactive inferencing system, as described above.

A. Simulated Networks

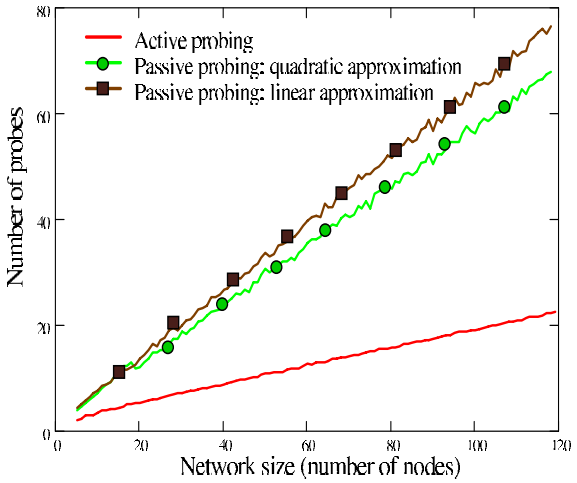
Our initial set of experiments was performed on randomly generated networks. Of course, those artificial networks may not necessarily reflect the realistic topologies, but they still provide an initial comparison of active probing to other approaches; then, the next section demonstrates active probing on real-life networks.

The artificial networks in our experiments were generated as follows. For each network size n, we generated twenty random networks with n nodes by randomly connecting each node to four other nodes. The probe stations are selected randomly. The probes follow the least-cost path from each probe station to each node.

The states to diagnose are any single node being down or no failure anywhere in the network. Each node has the same prior probability of failure, and there is no noise in the probe results. Note that in this case n probes are sufficient, because one can always use just one probe-station and probe every single node.



(a)



(b)

Fig. 9. Active versus pre-planned probing results for randomly generated networks: simulation results on (a) small-scale and (b) large-scale networks.

Exhaustive search is performed to find the true minimum size probe set. Then linear-time subtractive search and quadratic-time greedy search are used to find probe sets. Active probing algorithm is evaluated as follows. For each network, we simulate all possible fault scenarios (i.e., a fault at each node, and the no-failure situation), and compute an average number of active probes needed for diagnosis in this network. Finally, for every probing method, we average the results over all networks of given size and report them in Figure 9.

The results in Figure 9a (small-scale networks) indicate that the approximation algorithms for finding the smallest probe set perform well and are much closer to the true minimum set size than to the upper bound of n probes and also demonstrate the considerable improvement resulting from active probing when compared with pre-planned, or "passive", probing. In Figure 9b the approximation and active probing algorithms are extended to larger networks for which finding the true minimum is impractical. The active probing demonstrates

	# of nodes	# of probes	Pre-planned probes (exact)	Pre-planned probes (greedy)	Pre-planned probes (detect)	Active Probing: min	Active Probing: max	Active Probing: average	Savings % active vs exact
O1	24/19	22/22	15	15	8	3	8	4.9	67%
O2	43/38	44/32	27	27	17	3	17	7.7	72%
O3	40/34	29/29	24	24	16	3	16	7.5	69%
G1	10/8	14/8	5	5	3	3	4	3.2	36%
C1	149/50	27/27	27	27	27	4	27	10.8	60%
C2	122/46	27/27	24	24	24	4	24	9.8	61%
C3	56/32	27/23	17	17	14	4	13	6.7	71%
C4	34/23	27/22	12	13	10	3	10	5.5	75%

Fig. 10. Active probing results on several practical problems.

more than 60% improvement over the pre-planned probing.

B. Practical Applications

In Table 10, we report the results on several real probing applications. The problem $G1$ is a relatively small testbed for probe analysis, while the sets of problems $O1 - O3$ and $C1 - C4$ relate to networks supporting several different e-business applications, which include many servers and routers, and its performance and availability depends on a large number of software components (such as various databases, etc). For the purposes of high-level, overall network diagnosis, only a set of aggregate components is specified (e.g., a particular network 'cloud', or a firewall of a specific company are considered as diagnosable components). A set of probes was manually selected by an expert for the case of single fault localization. We ran a simple initial preprocessing on the dependency matrix in order to eliminate repeating probes and merge indistinguishable nodes, i.e. the nodes whose faults are indistinguishable given the original probe set. The first two column of Table 10 show the number of nodes and probes before and after the initial preprocessing (e.g., originally, problem $O2$ had 43 nodes and 44 probes, which was reduced by preprocessing to 38 an 32, respectively). The next two columns show the minimum number of probes found in a pre-planning phase by exhaustive and by greedy search, respectively. Then the next column shows the minimum number of probes (found in greedy way) necessary for fault detection only (i.e., simply the probes 'covering' all nodes). Finally, we show the minimum and the maximum number of probes required by active probing to diagnose a single fault, and the average such number over all possible faults.

Our approach was quite successful for these applications. For example, in problem $O3$ having 34 nodes and 29 probes, the probe-set selection algorithm found that the minimum number of probes required for single fault localization is only 24 probes, a saving of 17%. Approximation algorithms were optimal or nearly optimal: greedy search returned 24 probes, while subtractive search found 25 probes (we only show the results of greedy search in the table since it was always

superior to subtractive search). Finally, the most impressive results were obtained by active probing. The number of probes needed never exceeded 16 probes; on average, active probing required only 7.5 probes, versus 24 probes used by pre-planned probing, which yields savings of 69% (and of almost 74% if the initial probe set is considered). In most of the cases (except for a small testbed problem $G1$), active probing was saving from 60% to 75% probes if compared to pre-planned probing (and even more if compared to the initial probe set size).

VIII. Modelling Change in Dynamic Systems

Many commonly used approaches to diagnosis (including the *codebook* [7] and the active probing described above) assume that the system does not change during the diagnosis. Namely, the probe outcomes are obtained while the system is in a certain state that we wish to recover from the observations. However, this is not the case in highly dynamic systems where the failure and repair rates are comparable with average diagnosis time. In such cases, probes can provide contradictory information: e.g. same probe which failed a couple of minutes ago is OK now - does this mean a repair occurred (i.e., the failure was intermittent) or one of the probe outcome was incorrect (noisy probe outcomes)? In "static" approaches, such inconsistencies are typically treated as noise [7], and may (unnecessarily) increase the diagnostic error.

Another challenge is the presence of multiple safe faults which become more likely with the growing size of a dynamic system. "Static" approaches (e.g. codebook) do not track the sequential occurrence of the faults (and repairs), and thus face the problem of diagnosing *simultaneous* multiple faults present in the system. However, general multifault diagnosis problem is known to be computationally hard (e.g., constrained-satisfaction formulation of [22] and probabilistic inference problem in Bayesian networks [1] are both NP-hard; also, handling multiple faults in a system of n components using codebook [7] or active probing [3] would require enumeration of up to 2^n fault combinations).

Herein, we propose two approaches to handling dynamic systems. The first one is a general framework of *Dynamic Bayesian Networks (DBNs)*, that can handle a wide range of dynamic systems, but suffers from same computational complexity problem as the basic Bayesian network framework (i.e., inference in DBNs is NP-hard). The second one is an efficient linear-time approximation, called herein *sequential multifault* approach. This approach provides significant computational savings over general multifault approaches (such as BNs and DBNs), while at the same improves the accuracy of "static" approaches (such as codebook and single-fault active probing). The price to pay is the restricting assumptions that failures and repairs happen only one at a time and with a relatively low frequency so that the diagnostic engine can process them sequentially. However, there appears to be a wide range of practical diagnostic problems satisfying these assumptions, where our approach provides a nice trade-off between accuracy and complexity of diagnosis.

A. Dynamic Bayesian Networks

In order to model situations where the states of system components change over time, we will first consider general framework of Dynamic Bayesian Networks (DBNs). Dynamic BN model extends static BN model by introducing the notion of time, namely, by adding time slices and specifying transition probabilities between these slices: $P(X^t|X^{t-1})$, where $X^t = (x_1^t, ...x_n^t)$ is the vector of node states at time slice t. DBNs use the Markov assumption that the future system state is independent of its past states given the present state. Of course, a brute-force specification of such probability over binary variables requires a table of size $O(2^{2n})$, but the dynamic Bayesian network exploits the conditional independencies (just as regular BN) which allow to decompose the transition probability into a product of transition probabilities for each node at time slice t. As a result, a DBN is defined as a two-slice BN, where the intra-slice dependencies are described by a static BN, and inter-slice dependencies describe the transition probabilities. It is usually assumed that DBN describes a stationary stochastic process (i.e. the transition probabilities and intra-slice dependencies do not change with time), thus only two slices are enough to describe the process. Figure 11b shows an example of a dynamic BN that extends a static BN in Figure 11a (describing intra-slice dependencies) by adding inter-slice dependencies encoding transition probabilities. For example, the state of node X_1 at time $t+1$ depends on the states of nodes X_1 and X_2 at time t, which is described by a transition probability distribution $P(X_1^{t+1}|X_1^t, X_2^t)$. Node T_j^t denotes j-th probe outcome observed at time t; note that at different time slices, different probes can be observed (sometimes simultaneously, depending on the size of time slice with respect to probe time window).

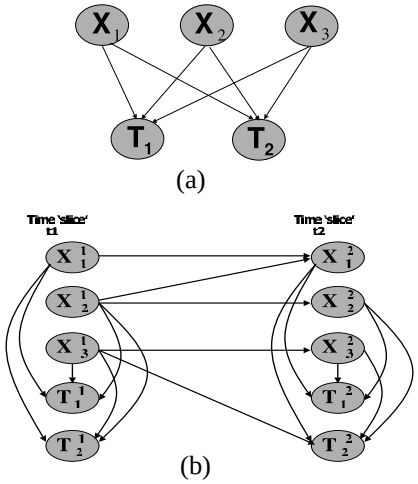


Fig. 11. (a) A two-layer Bayesian network structure for a set $\mathbf{X} = (X_1, X_2, X_3)$ of network elements and a set of probes $\mathbf{T} = (T_1, T_2)$, and (b) its extension to a Dynamic Bayesian Network.

Once a dynamic BN is specified, any standard Bayesian Network inference algorithm can be applied to the two-slice dynamic BN in order to compute $P(X^t|X^{t-1}, Y^t)$, given the prior distribution $P(X^{t-1})$ and the observations Y^t at time

slice t (where $Y \subseteq T$ is a subset of probes observed at time t). Clearly, this process can be repeated iteratively over time, always producing the joint distribution over the system states at the current time and the most-likely diagnosis. Although the computational complexity of exact inference in DBNs can be quite high, up to $O(2^n)$, where n is the number of nodes at a time slice, there are some efficient approximation algorithms available [23], [24].

1) Learning Dynamic Bayesian Networks: In order to apply dynamic Bayesian networks to our diagnostic problems we will need to obtain transition probability parameters, as well as the probe outcome probabilities (in case of noisy probes). Recall that we are only given the dependency matrix that defines the intra-slice structure; the inter-slice structure we assume to be quite simple: each node fails and repairs independently of the others, thus every node state at time $t-1$ affects only its state at the next time step, and vice versa, every node's state now is only affected by its own state on the previous time slice. However, the probability parameters must be learned from data, such as sequences of probe observations over several time slices.

A standard approach to learning DBNs from sequential data is to use Expectation-Maximization (EM) [25] algorithm which is a generic technique for learning probabilistic models with hidden variables (i.e. variables that are not observed, or missing in the data). The algorithm works iteratively as follows: it uses some initial parameter assignment to compute the expected values of certain sufficient statistics (e.g., frequencies in case of maximum-likelihood probability estimates) which cannot be computed directly due to missing values of certain variables (this is called the E-step); then the expected statistics are used instead of "real" ones (as if they were computed from the data without hidden variables) and either a maximum-likelihood approach or, alternatively, MAP - maximum-aposteriory probability approach which uses Bayesian priors over the parameters – is applied in order to compute updated parameters (M-step); the new parameters replace the ones from the first iteration, and the algorithm continues recursively. It is known to converge to a local maximum in parameter space under quite general assumptions.

2) Empirical results: In Figure 12 we report empirical results for one of our real-life probing problems. We constructed a dynamic Bayesian network as described above, using the dependency matrix for creating intra-slice dependencies, and for inter-slice dependencies assuming only links between the node at time t and same node at time $t+1$. We ran EM algorithm on the resulting network in order to learn its parameters. (The experiments were performed using the open-source MATLAB library called *Bayes Net Toolbox (BNT)* [26].) The diagnostic algorithm using DBN was evaluated for different levels of noise and fault probabilities at each node.

We simulated various sequences of faults, assuming that the probability of repair (i.e., switching from failed to OK state) between the time slices is $P(X_i^{t+1} = 0 | X_i^t = 1) = 0.5$ for each node X_i, and that each probe outcome has noise 0.01, a parameter of noisy-OR model described before. (Parameter $ncases$ denotes the number of training sequences used for learning DBN, in this case $ncases = 15$; for testing, we used a sequence of probe outcomes at 5 consecutive time slices). It was also assumed that at the initial time slice, there is a single node failure, and the experiments were performed for each such failure. The dependency matrix was selected so that in the absence of noise we could uniquely diagnose a single fault. This makes it easier to investigate the effect of increasing noise on the diagnostic performance.

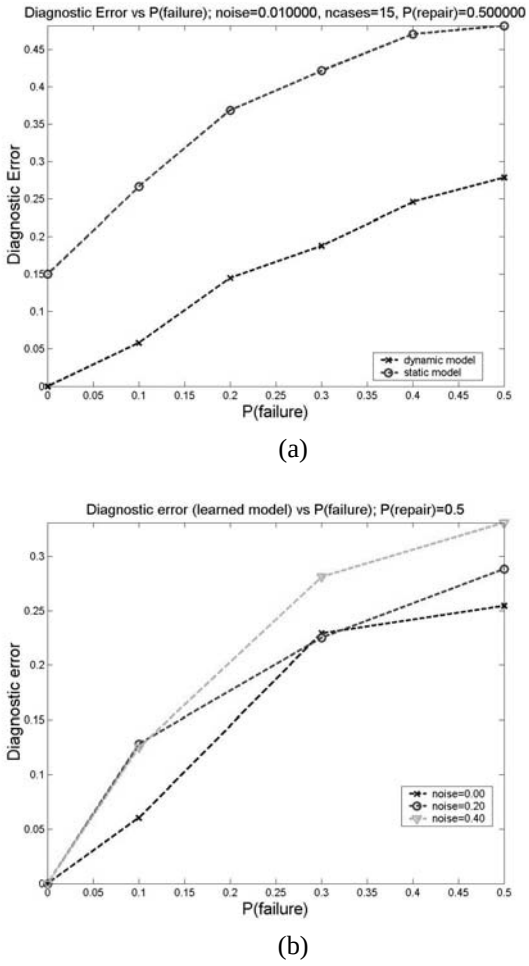


Fig. 12. Results of learning parameters of DBN for a practical probing application: (a) dynamic model (DBN) produces significantly more accurate diagnosis than the static BN model which assumes zero transition probabilities, i.e. no change in the system; also, both errors increase with increasing probability of fault at a node; (b) smaller noise level in probes yields, in general, higher diagnostic accuracy.

First, we compared the performance of learned DBN versus the static model that assumes no change in the system, i.e. zero transition probabilities: $P(X^{t+1}|X^t) = 0$ if $X^{t+1} \neq X^t$. In Figure 12a, the X-axis shows the probability of failure between the time slices, $P(X_i^{t+1} = 1 | X_i^t = 0)$, and the Y-axis shows the diagnostic error. We observe that the dynamic model produces much more accurate diagnosis (by 15-20%) than the static model. This observation confirms that using dynamic Bayesian networks instead of static ones is indeed beneficial when the system is dynamically changing.

We also observe that the diagnostic error of both static and dynamic models increases with the increasing probability of failure. Indeed, higher probability of failure implies more uncertainty about the node states, and more frequent state

change.

Figure 12b also shows how the diagnostic error increases with the (transitional) probability of failure, now for different levels of noise. As expected, we observe that the smaller noise levels generally yield a more accurate diagnosis. This corresponds to our intuition since more deterministic probes tend to keep more information about the true states of the nodes they go through; thus, learning uses more informative data.

B. Sequential Multifault Approach: an Efficient Approximation to DBNs

As we already mentioned above, the problem with DBNs is the computational complexity of inference that can be generally hard (NP-hard). Herein, we propose an approximation to DBNs based on simplifying assumptions regarding the fault occurrence: namely, we assume that failures and repairs happen one at a time, and the time between such events is sufficient for diagnostic engine to complete its cycle. Our approach can be also viewed as a very simple extension of the single-fault active probing algorithm (and it will be described in detail using the dependency matrix terminology).

Note that some multifault situations can be inherently unrecognizable because some components may become ”shielded” by the failures of other components. Particularly, in dependency matrix setting, we will define a component X as *shielded* by the failure of the component Y if all probes going through X go through Y as well.

Generic Multifault. First, we consider a very simple algorithm for handling multiple faults (called *generic multifault*) that still assumes no change in the system state during the diagnosis cycle. Given the dependency matrix and the probes outcomes, the algorithm

1. Finds OK nodes: these are all the nodes through which at least one OK probe passed.
2. Finds failed nodes: these are the nodes through which any failed probe passed such that all other nodes on its path are OK nodes, as determined in step 1.
3. Finds *shielded* nodes: these are the nodes through which no probe goes other than those that go through any of already failed nodes. Thus, all those probes will return ’failure’ and it is impossible to determine the state of such nodes, or, in other words, the nodes are ’shielded’ by the failures of nodes found in step 2.
4. The remaining nodes are ”possible failures”, in the sense that certain combinations of their failures can produce the given set of probe outcomes.

Generic multifault that reports shielded nodes as failed does not miss any faults, although its false-positive error (the amount of OK nodes reported as faulty) can be high for certain dependency matrices. We will refer to this algorithm as ”safe”, as opposed to ”non-safe” version that reports shielded nodes as OK. In fact, the generic multifault algorithm acknowledges the shielded nodes, and it is up to the user to decide how to interpret them. We will show in empirical section that, depending on the probability of fault in a system, we should prefer ”safe” or ”non-safe” version.

Sequential Multifault. We now extend the generic multifault approach to dynamic systems. The resulting algorithm, called *sequential multifault* is still linear in the number of probes, but has a lower diagnostic error because it keeps track of changes in the system: the inconsistencies in observations help to detect system change. The algorithm does not restrict the amount of faulty components in the system, but it assumes that only one change can happen at a time (i.e., failure or repair of one component), and that processing of each change is fast enough so that no other change occurs while the current change is being processed. At a very high-level, the algorithm performs the following monitoring loop:

```
initialize-system-state;
while (true)
    if  current observation contradicts
        previous observations {
               diagnose change;
               report results;
    }
```

Particularly, the algorithm monitors changes in the system’s states using two sets of probes: set for fix (i.e., repair) detection to monitor nodes that are known as failed, and set for failure detection to monitor nodes that are known to be OK. Just as for generic multifault, the algorithm has ”safe” and ”non-safe” ways of treating shielded nodes, which are compared in the empirical section.

If no change in the system has occurred, the probes from the first set are expected to continue returning ”failure”, whereas the probes from the second set are expected to continue returning OK. A probe outcome different from the one expected indicates a change in the system. When the algorithm detects a change, it diagnoses (locates) the changed component. Since the algorithm tracks the changes sequentially, it requires to be given an initial system state. If the initial system state is not known, it can be determined by applying the generic multifault algorithm. After a change has been located and processed, the algorithm updates its set of measurements - probe sets for fix and failure detection. It also determines the set of shielded nodes - the nodes that are shielded by the current set of failures. The pseudocode for the sequential multifault algorithm is shown in Figure 13.

1) Empirical Results: We compared the sequential and generic multifault algorithms versus single-fault active probing. The experiments were performed on randomly generated networks with 50 nodes and on a real-life router-level network topology. For synthetic networks, the probe sets were constructed by randomly choosing sources and then constructing shortest paths from each source to every other node. The number of probe stations varied from 1 to 10. The results were averaged over 30 trials for each number of sources. The simulations were run on sequences of 100 consecutive changes in the nodes’ states, which were generated randomly, one change at a time (i.e., at each step, only one node can change its state). At each step, the change type - failure or repair - was chosen according to a ”fault density” parameter that took values from 0 to 1 (fixed for each sequence, independent on the current system’s state), which allowed to vary the average

Sequential Multi-Fault algorithm

Input: Dependency matrix, probe outcomes as they arrive, initial system's state.
Output: diagnosed system state.
Initialize nodes according to the initial system state.
while (true) /* endless loop, i.e. continuous monitoring */
shielded nodes = shielded(set of current failures)
probesForFixDetection F = setForFixDetection()
probesForFailureDetection D = setForFailureDetection()
change = "no change"
while (change = "no change")
receive outcome of a probe P
if (outcome(P) = OK and P is in F)
change = "repair"
else if (outcome(P) = FAILED and P is in D)
change = "failure"
end
if (change = "repair")
move all nodes in P to OK nodes set
else if (change="failure")
fault = DiagnoseSingleFault(probe)
move fault to the failed nodes set
end
Output: OK nodes, failed nodes, shielded nodes
end

Fig. 13. Sequential Multifault Algorithm.

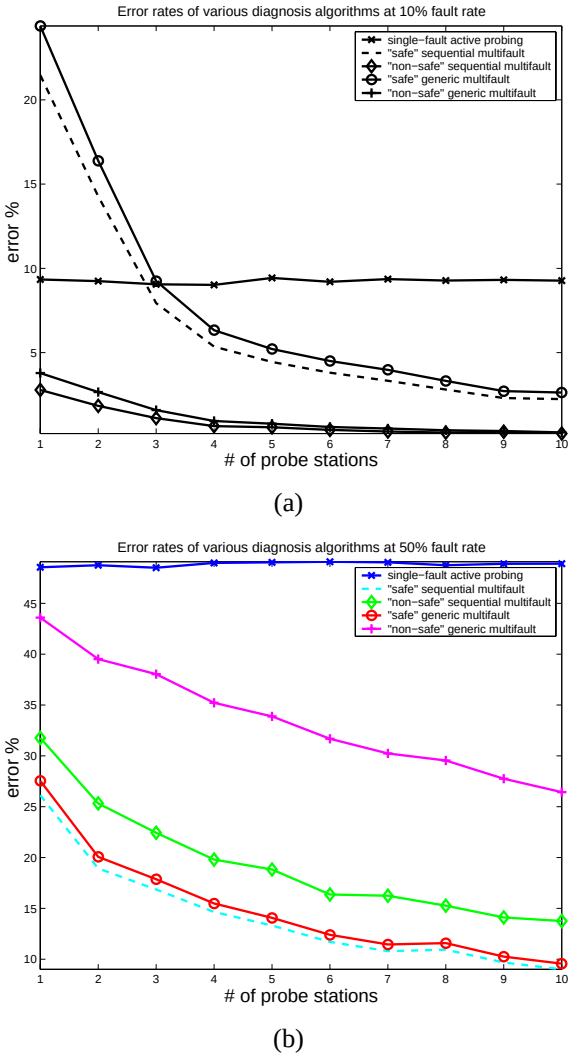


Fig. 14. Error rates (a) at 10% fault density and (b) at 50% fault density.

number of faults in the system over the whole sequence. We present results for the fault densities 10% and 50% (that is, when 10% and 50%, correspondingly, of all nodes were down on average).

We vary the number of probe stations, also called *sources*, and expect that increasing number of sources will decrease the amount of shielded nodes, since nodes may become reachable from multiple sources, and thus improve the accuracy of diagnosis. This indeed happens for multifault algorithms, while the single-fault one (active probing) is not affected – its error is always close to the fraction of faults in the system. Indeed, if there are N faults in the system, the single-fault algorithm will either find one of them, and miss all the others, or miss all of them, and instead diagnose some OK node as failure. In the first case, its false-negative error is N-1, with positive error being zero. In the second case, its false-negative error is N, and false-positive error is 1. So the lower bound on average total error is N-1, and the upper bound is N+1, where N is the number of actual faults in the system. This estimation holds for any implementation of the single-fault algorithm (active probing or codebook).

The figures 14a and 14b compare the algorithms for two different regimes, i.e. low and high fault density, and suggest that (1) multifault algorithms are generally more accurate than the single-fault one, especially for higher fault densities, and (2) selection of "safe" versus "non-safe" version depends on the anticipated fault density. When the fault density is relatively low (e.g. 10%), the "non-safe" sequential algorithm yields the best results (Figure 14a) with the total error rate more than 3 times less than that of the single-fault algorithm, even for the case of a single source. The error rate of the "safe" multifault algorithms is relatively high when the number of sources is small, but quickly decreases with the number of sources, also outperforming the single-fault algorithm. When the fault density is high (50% in Figure 14b), any of the multifault algorithms performs better than the single-fault algorithm (Figure 14b), with the "safe" versions yielding the best results. So at such fault rates it is better to interpret the shielded nodes as failed, rather than OK. It is interesting to note that, unlike low fault density, at high fault densities sequential "non-safe" multifault produces notably lower error than generic "non-safe" multifault algorithm. This difference can be explained by the fact that the sequential algorithm keeps track of the fault history, so if a node had failed and then later became shielded, sequential algorithm would "remember" this failure, rather than "blindly" declare all shielded nodes as OK, as the "non-safe" version of the generic multifault algorithm would do. For more detailed discussion and the results on real-life network (which looked quite similar to the simulated ones) see [27].

IX. Implementation: Real-time Diagnosis and Probe Selection Systems

The algorithms described above were used as a basis for building a system for probe set selection and dependency matrix analysis, called *DMA (Dependency Matrix Analysis tool)*, and a system for real-time diagnosis called *RAIL (Real-Time Active Inference and Learning)*.

A. DMA system

Given as an input a dependency matrix, the analyzer will first check for obviously redundant probes, e.g. delete identical copies of a probe (we encountered such cases in practice). DMA will also find all subsets of indistinguishable nodes and group them accordingly into meta-nodes. Then, DMA applies the following analysis to the resulting matrix:
- if exact search is not too expensive, DMA finds all optimal probe subsets using exact pre-planned probing;
- it finds a suboptimal probe subset using greedy search;
- it finds a probe set for problem detection only, using a greedy approach such as in active probing, until there are no 'uncovered' nodes left;
- DMA analyzes the effectiveness of active probing on the given dependency matrix by simulating a single fault, as well as no-fault situation, and computing the number of active probes required; it also computes an average number of active probes;
- finally, DMA computes the *induced width* of the Bayesian network corresponding to the dependency matrix; this parameter controls the complexity of exact inference which is exponential in the induced width. Interestingly, in most of our practical examples the induced width was quite low (6 to 8) while the Bayesian network contained up to 38 nodes and 32 probes (problem $O2$ described in the following section).

B. RAIL - diagnosis system

We used the active probing algorithm to implement the prototype real-time diagnostic system called RAIL. The system architecture is shown in Figure 15. The real-time diagnosis engine obtains the input through the Real-Time Event Manager (REM) which is a generic component able to process not only incoming probes but various other event types; thus the diagnostic engine is not probe-specific. In our particular application which uses the IBM's End-to-End Probing Platform (EPP), the input probe outcomes are obtained from the EPP probe stations, processed by REM, and submitted to the diagnostic engine which updates its beliefs about the system states, and requests an active probe, if needed, using the Probe Agent Driver component, which "talks" to EPP. The dependency matrix information is maintained and updated by Dependency Manager (DM), which obtains the initial matrix from the DMA tool. The future extensions to the system include adding a learning component that allows online adaptation of the dependency matrix to the changing environment (e.g., probe path changes, adding or deleting components, discovery of hidden causes, etc.), and learning parameters describing the dynamics of the system. In other word, the component will perform online learning of both structure and parameters of a DBN model.

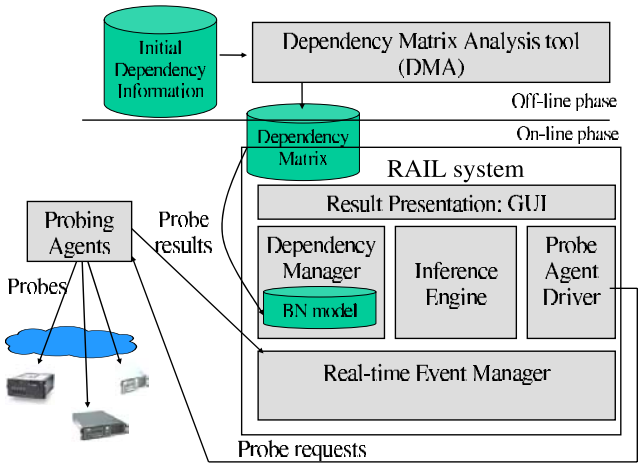


Fig. 15. RAIL system architecture.

In Figure 16 we show a sequence of screenshots of RAIL system running on a simple benchmark introduced in Figure 2. Each component corresponds to a node on the screen, and the *logical* links between the nodes are added in order to demonstrate how a probe can pass through a sequence of components, even though they are not physically connected: e.g., a link from WebServer (WS) to Application Server Hardware (HAS). The sequence of probe outcomes is shown in the upper-right window, while the results of diagnosis is shown in the lower-right window as a probability distribution over all nodes (in this simple demo, we assumed no more than one fault in a system). In this example, a Database Server failure occurred, which was diagnosed using the following sequence of active probes. Once 'main' probe (pWS) failed, the fault was detected, and the active probing selected $pDBS$ as a most-informative next probe. This probe was executed and returned a failure. Then $pingH$ (ping to 'hub' or 'router', thus a synonym for $pingR$) returned OK, thus the router was functioning properly; finally, $pingDBS$ returned OK which left as only possible fault DBS node. Thus, diagnosis completed using only 3 probes out of 7, and the 'main' probe that was run repeatedly for fault detection.

Recently, RAIL system was extended to diagnose sequential faults and repairs [4], [27] as described in the previous section. This approach is a bit more restrictive than general multi-fault diagnosis in Bayesian Networks and DBNs, as it still assumes a single failure at a time. However, this results into much more efficient practical approach that avoids possibly exponential complexity of multi-fault inference.

A scenario illustrating sequential multifault diagnosis is shown in Figure 17. In this example, we assume that there is no failed nodes in the initial system state.

1. *All nodes are functioning properly; all probes return OK* (Figure 17a).

Since there are no failures at the current state, probe set for fix detection is empty. Probe set for failure detection consists of the longest probe that covers all nodes, pWS.

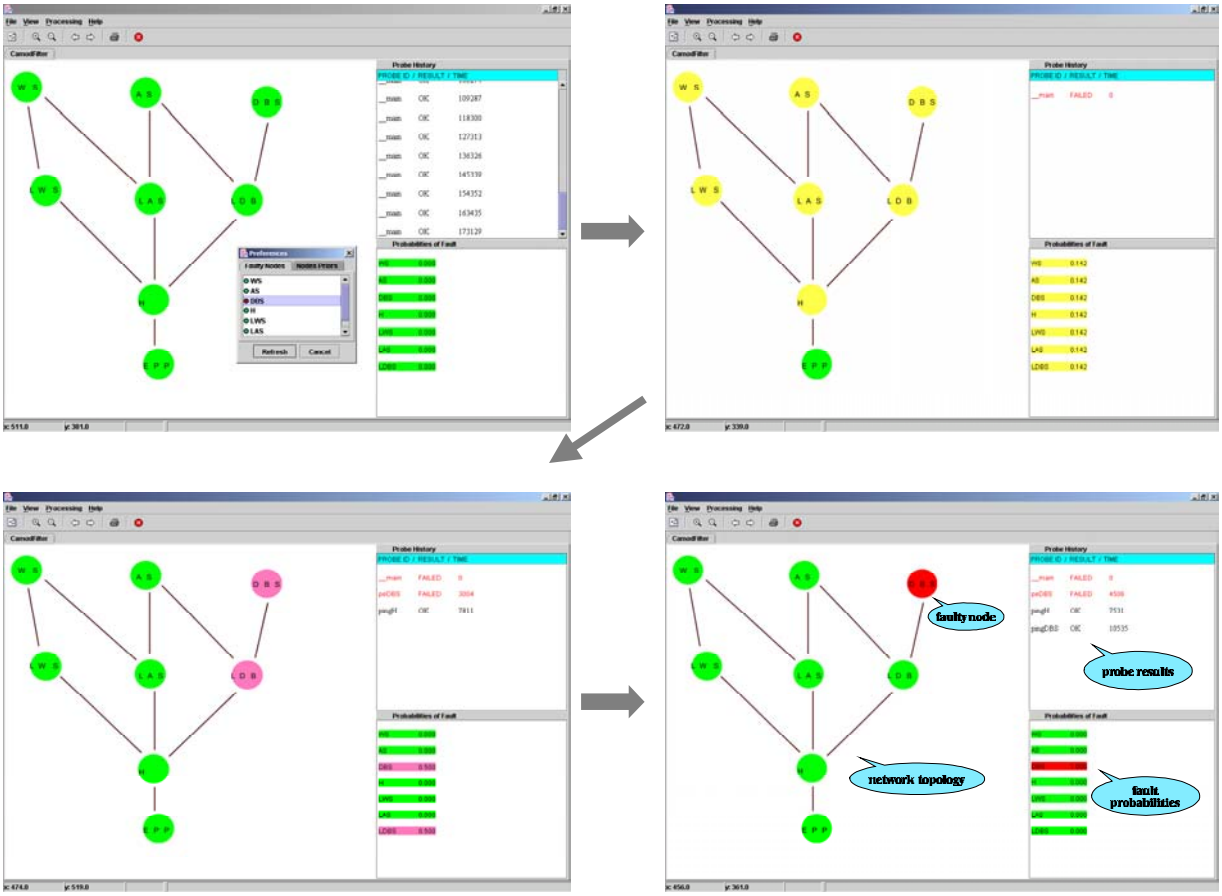


Fig. 16. A sequence of RAIL screenshots demonstrating successful diagnosis of a Database Server fault in our testbed system demonstrated in Figure 2.

2. *Node WS failed, probe pWS failed* (Figure 17b).
We locate the fault (WS) by active probing algorithm, update the diagnosed system's state, and delete WS and all probes going thorough it from the dependency matrix. We see that the failure of WS doesn't shield any nodes, because all remaining nodes still have some 1's in their columns.

3. *Node HAS failed, probe pAS returns failed* (Figure 17c).
After locating the fault (HAS) and modifying the dependency matrix, we see that HAS's failure has shielded node AS. Note also, that the failed node WS has also become shielded by this failure. So we don't include the probe pWS in the set for fix detection. Rather, we can check the status of WS after HAS has got repaired.

4. *Node HDBS failed, probe $pDBS$ failed* (Figure 17d).
The faulty node in this case is HDBS. Its failure shielded one more node - DBS. Probe set for failure detection consists of a single probe - $pingWS$ - that covers both of the remaining OK nodes (HWS and R). For fix, we monitor only HAS and HDBS; the failed node WS is shielded.

5. *Node HAS repaired, probe $pingAS$ returns OK* (Figure 17e).
Success of probe $pingAS$ indicates that HAS has repaired. However, we see that none of the shielded nodes became unshielded, because they all are still shielded by HDBS. Node WS is also shielded, so we don't include pWS into the probe set for fix detection.

6. *Node R failed, probe $pingWS$ returns failure* (Figure 17f).
Active probing finds that R is down. R's failure has shielded all the nodes. Now the probe set for failure detection is empty - we don't have any probe not going through a failed node. Probe set for fix detection consists of the single probe - $pingR$, that monitors node R for repair. We say that R is the bottleneck, because we cannot update information about other nodes' state before R has got fixed.

Note that some of the faults may "shield" the other nodes which become unreachable and thus impossible to diagnose (e.g., HAS failure makes AS unreachable). This indicates the necessity of developing more powerful probe sets that allow handling multiple faults. Note that even the exact multi-fault diagnosis is only as good as the input information provided by the probes.

X. Related Work

Previous work [9], [10], [28] studied the probe selection problem for the purpose of network management. This paper develops a more general framework for problem determination using probes, proves the NP-hardness of the probe-set

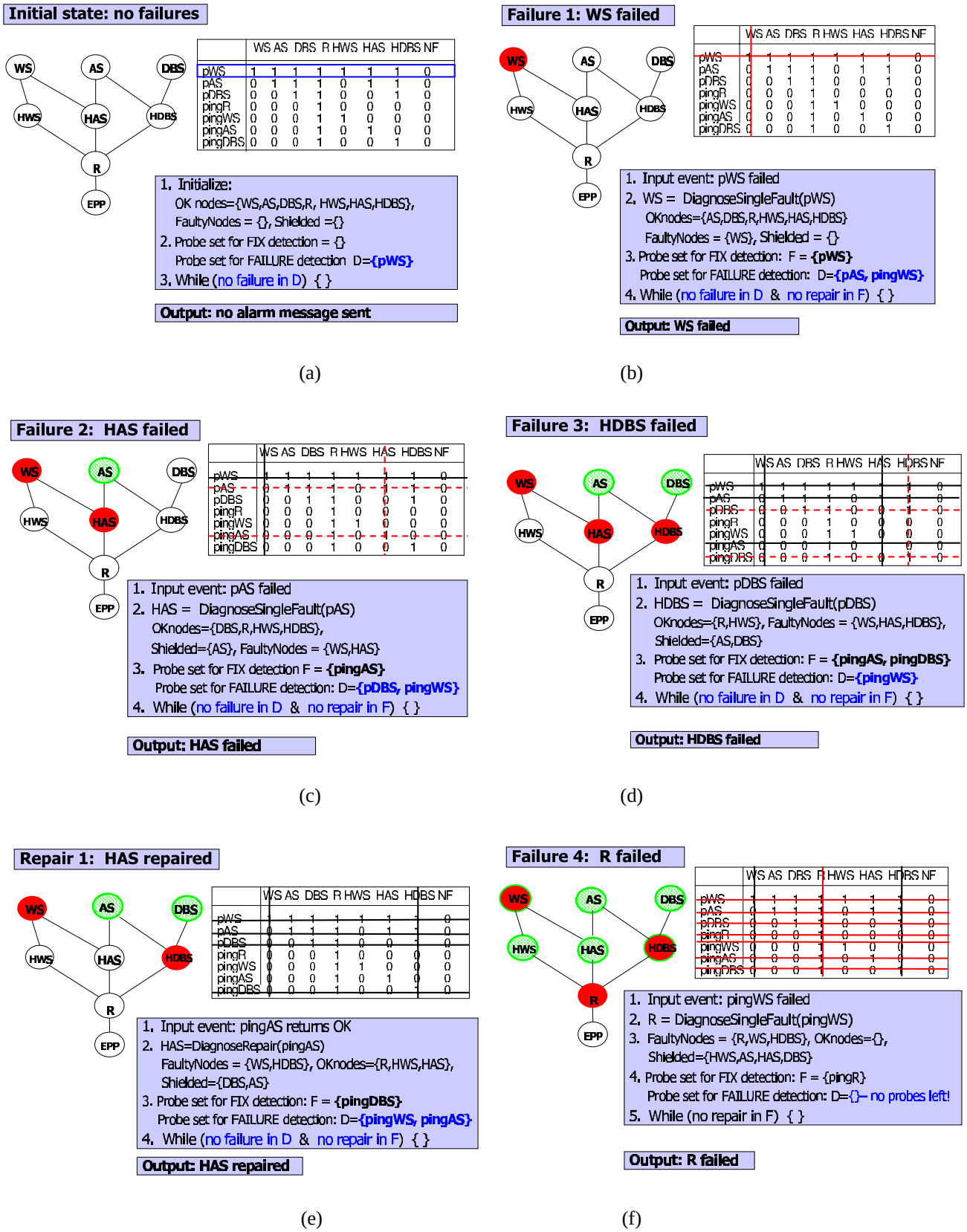


Fig. 17. Illustration of sequential multifault algorithm.

selection problem, develops the active probing approach and demonstrates its advantages in reducing probe-set size.

Event correlation [5], [6] for identifying root-causes has long been recognized as a critical issue in the system management domain. Problem determination is performed by analyzing alarms emitted by devices when a significant situation occurs. Unlike the probing scheme, alarms are "reactive" to a situation and this requires intensive instrumentation, only possible in a tightly managed environment. The probing approach uses test transactions that can be built easily without touching the existing devices.

Nonetheless, event correlation has many similarities to our work. The formulation of problem diagnosis as a "decoding" problem, where "problem events" are decoded from "symptom events", was first proposed by [7]. Our approach uses an active probing approach versus a "passive" analysis of symptom events; namely [7] selects codebooks (a combination of symptoms encoding particular problems) from a specified set of symptoms, while we actively construct those symptoms (probes), a much more flexible approach. Another important difference is that [7] lacks a detailed discussion of efficient algorithms for constructing optimal codebooks.

Other approaches to fault diagnosis in communication networks and distributed computer systems include Bayesian networks [29], [1] and other probabilistic dependency models [30]; another approach is statistical learning to detect deviations from the normal behavior of the network [31]. However the probe selection problem and active probing as formulated here were not addressed in that work. A comprehensive treatment of multi-fault, including most-informative test selection, was also provided in [22]. However, to the best of our knowledge, combining probabilistic inference with online selection of most-informative probes appears to be a novel approach in the area of distributed systems diagnosis via end-to-end probing.

In our recent work [1], [10], a theoretical bound on the diagnostic error was derived; this lower bound provides necessary conditions for the number of probes needed to achieve an asymptotically error-free diagnosis as the network size increases, given prior fault probabilities and a certain level of noise in probe outcomes. Also, empirical observations suggested that a simple greedy approximation algorithm for finding most-likely diagnosis (MPE) yield nearly-optimal results when noise is sufficiently small. In this paper, we provide a theoretical justification of these results.

XI. Conclusions

This paper addresses real-time problem diagnosis in computer networks and distributed computer systems. We improve the state-of-art by introducing a more adaptive and cost-efficient technique, called *active probing*. Active probing uses an information-theoretic approach to probe selection that speeds up real-time diagnosis by minimizing set of measurements, such as probes (test transactions), while maintaining high diagnostic accuracy.

We compare the existing pre-planned probe-selection approach [9], [10] with the active probing. In the pre-planned approach, a set of probes is selected and scheduled to run periodically. Problem diagnosis is performed by analyzing the probe results. This approach may be quite inefficient since it requires running an unnecessarily large set of probes capable of diagnosing *all* possible problems, many of which might in fact never occur. We prove that both problems of optimal probe set selection for fault detection and for fault diagnosis are NP-hard. We then review (and reformulate in information-theoretic framework) simple approximation techniques of [9], [10] that serve as a baseline for comparison with the proposed active probing approach.

In active probing, a set of probes is selected to run periodically as before, but only for the purpose of **detecting** when a problem occurs. Whenever occurrence a problem is detected by one or more of the probes, additional probes are sent out to obtain further information about the problem. As probe results are received, belief about the system state is updated using probabilistic inference in a Bayesian network, and the next most-informative probe is selected and sent. This process continues until the problem is diagnosed. We demonstrate through both analysis and simulation that the active probing scheme greatly reduces both the number of probes (by up to 75% in our real-life applications and in simulations) and the time needed for localizing the problem when compared with non-adaptive, pre-planned probing schemes.

We discuss probabilistic inference approach to analyzing probe results: a simple one based on k-fault assumption, and a generic multi-fault approach; we also provide theoretical guarantees on diagnostic quality of a simple approximate inference algorithm in the presence of noise in probe outcomes. These encouraging results suggest the applicability of such approximations to certain almost-deterministic diagnostic problems that often arise in practical applications. Moreover, we demonstrate initial results on learning parameters of Dynamic Bayesian Network from probe outcomes that allows online adaptation of diagnostic engine to system's dynamics. Since inference in DBNs is generally intractable, we also present an efficient approximate approach called sequential multifault that works well under certain restricting assumptions (such as single-node failure or repair in a system at a time); empirical results demonstrate clear advantage of such approaches over "static" techniques that do not handle system's changes. Finally, we describe an architecture and functionality of our proof-of-concept system that implements our approach.

Directions for future work include incorporating Dynamic Bayesian Network model into active probing in order to better handle system dynamics, such as intermittent faults, dynamic routing and lack of precise knowledge of the probe path, and adapting to non-stationary behavior of the system using online learning that yields a dynamic, adaptive, probing strategy. We feel that adaptive, cost-effective techniques for problem determination will become increasingly important if new-generation IT systems are to be capable of self-management and self-repair.

Acknowledgements

We would like to thank Vittorio Castelli, Daniel Oblinger and other members of the Machine-Learning group for many

insightful discussions that contributed to the ideas of this paper. We also thank EPP team, especially Herb Lee, Andy Frenkiel, and Marius Sabbath, for their collaboration on RAIL/EPP framework, and Douglas Griswold, Luis Moss, and Justin Ellis for providing the data and useful information about various practical issues related to probing.

REFERENCES

[1] I. Rish, M. Brodie, and S. Ma, "Accuracy vs. Efficiency Trade-offs in Probabilistic Diagnosis," in *Proceedings of AAAI-2002, Edmonton, Alberta, Canada*, 2002.

[2] M. Brodie, I. Rish, S. Ma, and N. Odintsova, "Active probing strategies for problem diagnosis in distributed systems," in *Proceedings of the The Eighteenth International Joint Conference on Artificial Intelligence (IJCAI-03), Acapulco, Mexico*, 2003.

[3] I. Rish, M. Brodie, N. Odintsova, S. Ma, and G. Grabarnik, "Real-time Problem Determination in Distributed Systems using Active Probing," in *Proceedings of 2004 IEEE/IFIP Network Operations and Management Symposium (NOMS 2004), Seoul, Korea*, 2004.

[4] N. Odintsova, I. Rish, and S. Ma, "Multifault Diagnosis in Dynamic Systems," in *Integrated management (IM-2005), Nice, France*, 2005.

[5] A. Leinwand and K. Fang-Conroy, *Network Management: A Practical Perspective, 2nd Edition*. Addison-Wesley, 1995.

[6] B. Gruschke, "Integrated Event Management: Event Correlation Using Dependency Graphs," in *DSOM*, 1998.

[7] S. Kliger, S. Yemini, Y. Yemini, D. Ohsie, and S. Stolfo, "A coding approach to event correlation," in *Intelligent Network Management (IM)*, 1997.

[8] A. Frenkiel and H. Lee, "EPP: A Framework for Measuring the End-to-End Performance of Distributed Applications," in *Proceedings of Performance Engineering 'Best Practices' Conference, IBM Academy of Technology*, 1999.

[9] M. Brodie, I. Rish, and S. Ma, "Optimizing probe selection for fault localization," in *Distributed Systems Operation and Management*, 2001.

[10] I. Rish, M. Brodie, and S. Ma, "Intelligent probing: a Cost-Efficient Approach to Fault Diagnosis in Computer Networks," *IBM Systems Journal*, vol. 41, no. 3, pp. 372–385, 2002.

[11] R. M. Karp, *Complexity of Computer Computations*. Plenum Press, 1972, ch. Reducibility among combinatorial problems, pp. 85–103.

[12] M. Garey and D. Johnson, *Computers and Intractability; A Guide to the Theory of NP-completeness*. W.H. Freeman and Co., San Francisco, 1979.

[13] T. Cover and J. Thomas, *Elements of information theory*. New York:John Wiley & Sons, 1991.

[14] J. Pearl, *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, 1988.

[15] G. Cooper, "The computational complexity of probabilistic inference using Bayesian belief networks," *Artificial Intelligence*, vol. 42, no. 2–3, pp. 393–405, 1990.

[16] S. Lauritzen and D. Spiegelhalter, "Local computation with probabilities on graphical structures and their application to expert systems," *Journal of the Royal Statistical Society, Series B*, vol. 50(2), pp. 157–224, 1988.

[17] N. Zhang and D. Poole, "Exploiting causal independence in Bayesian network inference," *Journal of Artificial Intelligence Research*, vol. 5, pp. 301–328, 1996.

[18] R. Dechter, "Bucket elimination: A unifying framework for probabilistic inference," in *Proc. Twelfth Conf. on Uncertainty in Artificial Intelligence*, 1996, pp. 211–219.

[19] R. Dechter and I. Rish, "A scheme for approximating probabilistic inference," in *Proc. Thirteenth Conf. on Uncertainty in Artificial Intelligence (UAI97)*, 1997.

[20] ——, "Mini-buckets: A General Scheme for Approximating Inference," *J. of ACM*, no. 50(2), pp. 107–153, 2003.

[21] J. Yedidia, W. T. Freeman, and Y. Weiss, "Generalized belief propagation," in *NIPS 13*. MIT Press, 2001, pp. 689–695.

[22] J. de Kleer and B. Williams, "Diagnosing Multiple Faults," *Artificial Intelligence*, vol. 32, no. 1, 1987.

[23] K. Murphy and Y. Weiss, "The factored frontier algorithm for approximate inference in dbns," in *Proceedings of UAI-2001*, 2001.

[24] X. Boyen and D. Koller, "Tractable inference for complex stochastic processes," in *Proceedings of the 14th Annual Conference on Uncertainty in AI (UAI-98)*, 1998, pp. 33–42.

[25] A. Dempster, N. Laird, and D. Rubin, "Maximum Likelihood from Incomplete Data via The EM Algorithm," *Journal of Royal Statistical Society*, vol. 39, pp. 1–38, 1977.

[26] K. Murphy, "Bayes net toolbox for matlab," *http://www.ai.mit.edu/ murphyk/Software/BNT/bnt.html*.

[27] N. Odintsova, I. Rish, and S. Ma, "Multifault Diagnosis in Dynamic Systems," IBM T.J. Watson Research Center, Tech. Rep. RC23385, 2004.

[28] H. Ozmutlu, N. Gautam, and R. Barton, "Zone recovery methodology for probe-subset selection in end-to-end network monitoring," in *Network Operations and Management Symposium*, 2002, pp. 451–464.

[29] J. Huard and A. Lazar, "Fault isolation based on decision-theoretic troubleshooting," Center for Telecommunications Research, Columbia University, New York, NY, Tech. Rep. 442-96-08, 1996.

[30] I.Katzela and M.Schwartz, "Fault identification schemes in communication networks," in *IEEE/ACM Transactions on Networking*, 1995.

[31] C. Hood and C. Ji, "Proactive network fault detection," in *Proceedings of INFOCOM*, 1997.

APPENDIX A

Here we prove that PROBE-SET SELECTION FOR FAULT DIAGNOSIS is NP-hard.

Proof: We will show that PROBE-SET SELECTION FOR FAULT DIAGNOSIS is NP-hard via a reduction from PROBE-SET SELECTION FOR FAULT DETECTION.

Let the dependency matrix $D_{\mathbf{T},\mathbf{F}}$ and the positive integer k be an arbitrary instance of PROBE-SET SELECTION FOR FAULT DETECTION. We need to construct, in polynomial time, sets $\hat{\mathbf{F}}$, $\hat{\mathbf{T}}$, and an integer $\hat{k} \leq |\hat{\mathbf{T}}|$, such that there exists a subset of $\mathbf{T}$ of size at most k covering $\mathbf{F}$ **if and only if** there exists a subset of $\hat{\mathbf{T}}$ of size at most $\hat{k}$ such that $D_{\hat{\mathbf{T}},\hat{\mathbf{F}}}$ has unique columns.

We assume there are m possible faults, i.e. $\mathbf{F} = \{F_1, \ldots, F_m\}$. The set $\hat{\mathbf{F}}$ contains all the elements of $\mathbf{F}$ plus m new elements $G_1, \ldots, G_m$. The set $\hat{\mathbf{T}}$ will contain all the probes in $\mathbf{T}$, plus m additional probes $\{F_i, G_i\}$ for all $i \in \{1, \ldots, m\}$. The resulting dependency matrix will be of size $(r+m)\times(2m)$. It can be visualized as the four-block matrix

$$\begin{bmatrix} D_{\mathbf{T},\mathbf{F}} & 0 \\ I_m & I_m \end{bmatrix},$$

where I_m denotes the $m \times m$ identity matrix. We also set $\hat{k} = k + m - 1$ (for reasons that will become clear later). It is easy to see that the reduction can be done in polynomial time. We now show both the sufficient and necessary conditions.

For the "if" direction, we assume that $\langle D_{\mathbf{T},\mathbf{F}}, k\rangle \in$ PROBE-SET SELECTION FOR FAULT DETECTION, thus there exists a subset $\mathbf{T}' \subseteq \mathbf{T}$ of size at most k such that $D_{\mathbf{T}',\mathbf{F}}$ has at least one 1 in every column. We need to show that $\langle D_{\hat{\mathbf{T}},\hat{\mathbf{F}}}, \hat{k}\rangle$ is a positive instance of PROBE-SET SELECTION FOR FAULT DIAGNOSIS. Indeed, there exists a subset $\mathbf{Q}$ of at most rows $\hat{k}$ rows inducing a submatrix of $D_{\hat{\mathbf{T}},\hat{\mathbf{F}}}$ with unique columns – namely, $\mathbf{Q}$ contains $\mathbf{T}'$ and any $(m-1)$ of the m additional rows $\{F_i, G_i\}$. Since $D_{\mathbf{T}',\mathbf{F}}$ has a 1 in every column, every column of $D_{\mathbf{Q},\hat{\mathbf{F}}}$ corresponding to a fault in $\mathbf{F}$ will differ from any column corresponding to a fault in $\{G_1, \ldots, G_m\}$. Since removing a single row from I_m leaves all the m columns distinct, the columns corresponding to the elements of $\mathbf{F}$ will all be distinct from each other, as are the columns corresponding to the elements of $\{G_1, \ldots, G_n\}$. Notice that $|\mathbf{Q}| \leq k+m-1 = \hat{k}$, as desired.

To show the other direction, we assume that $\langle D_{\mathbf{T},\mathbf{F}}, k\rangle$ is a negative instance of PROBE-SET SELECTION FOR FAULT DETECTION. We need to show that $\langle D_{\hat{\mathbf{T}},\hat{\mathbf{F}}}, \hat{k}\rangle$ will be a negative instance of PROBE-SET SELECTION FOR FAULT DIAGNOSIS. Indeed, if any subset of at most k rows of $D_{\mathbf{T},\mathbf{F}}$ induces an all-0 column, then any submatrix of $D_{\hat{\mathbf{T}},\hat{\mathbf{F}}}$ induced by a subset of at most $\hat{k}$ rows will contain duplicate columns. To see this, let $\mathbf{Q}$ be a subset of at most $\hat{k}$ rows. There are two cases to be considered:

1) $\mathbf{Q}$ contains at most k elements of $\mathbf{T}$. Then, since there is an all-0 column in $D_{\mathbf{T},\mathbf{F}}$ induced by $\mathbf{Q}\cap\mathbf{T}$, we know that $D_{\mathbf{Q},\hat{\mathbf{F}}}$

will have identical columns corresponding to faults F_i and G_i, where i is the index of the all-0 column in $D_{\mathbf{T},\mathbf{F}}$.

2) $\mathbf{Q}$ contains more than k elements of $\mathbf{T}$. In this case $\mathbf{Q}$ cannot contain more than $m-2$ of the new rows (since $|\mathbf{Q}| \leq \hat{k}$). But then $\mathbf{Q}$ will induce duplicate columns in the right half of $D_{\hat{\mathbf{T}},\hat{\mathbf{F}}}$, since removing at least two rows from I_m induces duplicate columns.

Thus any subset $\mathbf{Q}$ of at most $\hat{k}$ rows induces duplicate columns in $D_{\hat{\mathbf{T}},\hat{\mathbf{F}}}$, making it a negative instance of PROBE-SET SELECTION FOR FAULT DIAGNOSIS, as desired.

■

APPENDIX B

Here we derive the equation 4 for $I(X;T)$. Here $X \in \{0,1,...,n\}$ denotes the state of the system under the single-fault assumption, where $X=0$ corresponds to the no-fault situation, and $X=i$ corresponds to failure of node X_i. A particular prior distribution $P(X)$ is assumed where $P(X=0)=1-p$, and the probability of fault in the system, p, is uniformly distributed over the nodes, i.e. $P(X=i)=p/n$ where $1 \leq i \leq n$. We assume that T is a probe of length k, and that $T=0$ denotes probe's success, while $T=1$ denotes probe's failure. Note that $I(X;T) = H(X) - H(X|T)$, where

$$\begin{aligned} H(X) &= -(1-p)\log(1-p) - n\frac{p}{n}\log\frac{p}{n} = \\ &= -(1-p)\log(1-p) - p\log\frac{p}{n}, \quad and \end{aligned} \tag{9}$$

$$\begin{aligned} -H(X|T) &= \sum_{t,x} P(x,t)\log P(x|t) = \\ &= \sum_{t} P(t)\sum_{x} P(x|t)\log P(x|t). \end{aligned} \tag{10}$$

Due to single-fault assumption, a probe T of length k fails if *exactly* one of its nodes fail, yielding a union of k mutually exclusive node failure events, each having probability p/n. Therefore,

$$P(T=1) = \frac{kp}{n}, \ and \ P(T=0) = 1 - \frac{kp}{n} = \frac{n-kp}{n}. \tag{11}$$

Next,

$$\begin{aligned} P(X=0|T=1) &= 0, \\ P(X=0|T=0) &= \frac{P(X=0,T=0)}{P(T=0)} = \frac{P(X=0)}{P(T=0)} = \\ &= \frac{1-p}{1-kp/n} = \frac{n(1-p)}{n-kp}, \end{aligned} \tag{12}$$

since $X=0$ is the intersection of events $X=0$ and $T=0$ (i.e., the event $X=0, T=0$).

Let us now compute $P(X=i|T)$ for $1 \leq i \leq n$, and $T \in \{0,1\}$. We will consider two cases: when X_i belongs to the probe T, and when it does not. We will abuse the notation slightly, and say $X_i \in T$ if X_i is on the path of probe T. Then, we can compute $P(X=i|T) = P(X=i, X_i \in T|T) + P(X=i, X_i \notin T|T)$ as follows:

$$\begin{aligned} P(X=i, X_i \in T|T=0) &= \frac{P(X=i, X_i \in T, T=0)}{P(T=0)} = 0, \\ P(X=i, X_i \notin T|T=0) &= \frac{P(X=i, X_i \notin T, T=0)}{P(T=0)} = \\ &= \frac{P(X=i)}{P(T=0)} = \frac{p}{n-kp}, \end{aligned} \tag{13}$$

since the intersection of events $X=i, X_i \in T, T=0$ is empty (incompatible events), and the intersection of events $X=i, X_i \notin T, T=0$ is simply $X=i$. Similarly,

$$\begin{aligned} P(X=i, X_i \in T|T=1) &= \frac{P(X=i, X_i \in T, T=1)}{P(T=1)} = \\ = \frac{P(X=i)}{P(T=1)} &= \frac{p/n}{kp/n} = 1/k, \\ P(X=i, X_i \notin T|T=1) &= \frac{P(X=i, X_i \notin T, T=1)}{P(T=1)} = 0, \end{aligned}$$

since the intersection of events $X=i, X_i \in T, T=1$ is simply $X=i$, and the intersection of events $X=i, X_i \notin T, T=1$ is empty (incompatible events).

Now, we can combine the equations above to compute the negative conditional entropy in 10. We decompose $P(X|T)$ into sum over a set mutually exclusive events, (1) $X=0$, and (2) $\{X=i, i>0\}$, where each event $X=i, i>0$ can be further partitioned into $X=i, X_i \in T, i>0$ and $X=i, X_i \notin T, i>0$. As the result, we get the following expression for $-H(X|T)$:

$$\sum_{t\in\{0,1\}} P(T=t) \sum_{i\in\{0,1,...,n\}} P(X=i|T=t)\log P(X=i|T=t) =$$

$$= \sum_{t\in\{0,1\}} P(T=t)\big[P(X=0|T=t)\log P(X=0|T=t) +$$

$$+ \sum_{i\in\{1,...,n\}} \big[P(X=i, X\in T|T=t) +$$

$$+ P(X=i, X\notin T|T=t)\big]\log P(X=i|T=t)\big]\big] =$$

$$= \frac{n-kp}{n}\big[\frac{n(1-p)}{n-kp}\log\frac{n(1-p)}{n-kp} + k\cdot 0 +$$

$$+ (n-k)\frac{p}{n-kp}\log\frac{p}{n-kp}\big] +$$

$$+ \frac{kp}{n}\big[0 + k\frac{1}{k}\log\frac{1}{k} + (n-k)\cdot 0\big] =$$

$$= (1-p)\log\frac{1-p}{1-k\frac{p}{n}} + (n-k)\frac{p}{n}\log\frac{p}{n-kp} - k\frac{p}{n}\log k. \tag{14}$$

Substituting the expression 14 for $-H(X|T)$ in 10 and combining it with th the equations 9, we get exactly the equation 4:

$$I(X;T) = I(p,n,k) = -(1-p)\log(1-p) - p\log\frac{p}{n}$$
$$-k\frac{p}{n}\log k + (1-p)\log\frac{1-p}{1-k\frac{p}{n}} + (n-k)\frac{p}{n}\log\frac{p}{n-pk}.$$