Computationally Efficient Tiered Inference for Multiple Fault Diagnosis

Juan Liu, Lukas Kuhn, and Johan de Kleer
Palo Alto Research Center,
Palo Alto, CA 94304 USA
{jjliu, lkuhn, dekleer}@parc.com

Abstract

Diagnosing multiple-component systems is difficult and computationally expensive, as the number of fault hypotheses grows exponentially with the number of components in the system. This paper describes an efficient computational framework for statistical diagnosis featuring two main ideas: (1) structuring fault hypotheses into tiers, starting from low cardinality fault assumptions (e.g., single fault) and gradually escalating to higher cardinality (e.g., double faults, triple faults) when necessary; (2) at each tier, dynamically partitioning the overall system into subsystems, within which there is likely to be a single fault. The partition is based on correlation between the system components and is dynamic: when a particular partition is ruled out, a new one is constructed based on the updated belief. When no viable partition remains, the search proceeds to the next tier. This approach enables the use of single-fault diagnosis, which has only linear complexity, to the subsystems avoiding exponential hypothesis explosion. We demonstrate the concepts and implementation via examples and simulation. We analyze the performance and show that for practical systems where most components are functioning properly, the proposed scheme achieves a desirable tradeoff between computational cost and diagnosis accuracy.

1 Introduction

Troubleshooting a practical system to isolate broken components can be difficult, as the number of fault combinations grows exponentially with the number of components. In diagnosis literature, various ideas have been proposed to address the computational challenge. The general diagnosis engine (GDE) work (de Kleer and Williams 1987) finds minimal diagnoses, isolating not the complete fault combination, but a minimal subset of broken components that can explain the observations. Another example is the production plant diagnosis work (Kuhn and de Kleer 2008), which extends model-based diagnosis (Reiter 1987; de Kleer and Williams 1987) to production systems such as food processing plants, oil refineries, and printers. The diagnosis engine (Kuhn and de Kleer 2008) discriminates fault assumptions based on their complexity. Diagnosis starts with simple fault assumptions (e.g., single, persistent, and/or independent faults) for computationally efficient diagnosis, and escalates to more complicated fault assumptions (e.g., multiple, intermittent, and/or interaction faults) when necessary. This progression of diagnosis greatly reduces computation complexity.

The minimal diagnosis idea and the progressive diagnosis work are qualitative in nature. Can we extend similar ideas from qualitative reasoning to statistical inference? Can we perform Bayesian updates in a computationally efficient manner? This are the questions we address in this paper.

Statistical inference is now widely adopted in diagnosis. The basic idea is to evaluate hypotheses (fault combinations) based on their probability given the observation data (Berger 1995). Mathematically, for any hypothesis $\mathbf{x}$ in the hypotheses space $\mathcal{X}$, we update its probability via the Bayes rule:

$$p(\mathbf{x}|o) = \alpha p(o|\mathbf{x})p(\mathbf{x}), \tag{1}$$

where $p(\mathbf{x})$ is the initial probability (prior) for the hypothesis $\mathbf{x}$, $p(o|\mathbf{x})$ is the likelihood probability of observing o given that $\mathbf{x}$ is true, and α is the normalization factor to let $p(\mathbf{x}|o)$ sum up to 1. The resulting $p(\mathbf{x}|o)$ is the posterior probability that $\mathbf{x}$ is true given the observation o. The diagnosis that best explains the data is the maximum a posterior (MAP) estimate

$$\mathbf{x}_{MAP} = \arg\max_{\mathbf{x}\in\mathcal{X}} p(\mathbf{x}|o). \tag{2}$$

While Bayesian update offers a coherent and quantitative way of incorporating observation data, it faces the same need to search through all hypotheses in $\mathcal{X}$. In practice, a system with M components has the hypothesis space

$$\mathcal{X} = \{000000, 000001, \ldots, 111111\}$$

Each hypothesis $\mathbf{x} \in \mathcal{X}$ is a bit vector, where i-th bit is an indicator whether the i-th component has fault (0 for not having fault, 1 for having fault). The computational complexity of the Bayesian update is $O(2^M)$. When M is large, the update is prohibitively expensive.

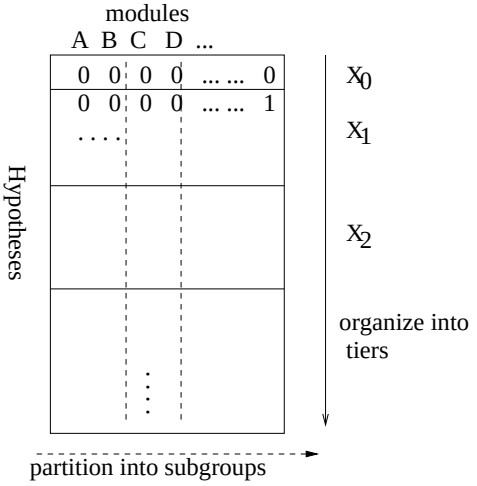

Figure 1: Basic idea: (1) organize hypothesis into tiers (along the vertical direction), and (2) partition components into subgroups (along the horizontal direction), for instance, AB are a group, and CD are a group.

In this paper, we propose two ideas to mitigate the computational difficulty. The first is tiered inference, illustrated in Sec. 2. The basic idea is to organize the hypothesis space $\mathcal{X}$ into tiers with increasing fault cardinality. Inference is restricted to lower tiers (fewer defective modules) until the lower tiers have been ruled out by the observation data. This idea is implicit in many diagnosis engines such as GDE (de Kleer and Williams 1987) and MBD (Thiebuax et al. 1996), but we develop it as part of a more general framework. Our main contribution is the second idea: a divide-and-conquer strategy presented in Sec. 3. It partitions system components into single-fault subsystems. This partitioning enables utilizing single-fault diagnosis, which only has linear complexity, to diagnosing a multiple-fault system.

Figure 1 illustrates these basic concepts. In the diagram, the hypothesis space $\mathcal{X}$ is represented as a matrix, with columns representing components, and rows representing the different fault assumptions. Organizing hypotheses into tiers is shown as dividing the hypothesis into vertically stacked blocks. Inference starts from the top block (no-fault tier), and progresses down to the single-fault tier $\mathcal{X}_1$, then to the double-fault tier $\mathcal{X}_2$, and so on. The second idea is to organize modules into groups, for instance, AB form a subsystem, and CD forms a subsystem. This forms a horizontal partition in the figure.

While partitioning a multiple-fault system into single-fault subsystems is a neat idea, how to partition is actually a tricky problem. We take a best-effort approach: given the posterior belief $\{p(\mathbf{x})\}$, we seek a partition which results in subsystems that are single-fault with maximum probability. Sec. 4 describes a computationally efficient greedy algorithm based on the intuition that modules within a subsystem must be negatively correlated so that the total number of faults remains constant (single-fault). The partitioning idea and algorithm are the main novelty of this paper.

Many diagnosis approaches have taken advantage of the hierarchical structure of the system being diagnosed (Pravan 2001) (Srinivas 1994). For example, all possible combinations of faults in a subsystem can be represented as a single component, as done in (Siddiqi and Huang 2007). Similarly, if two distinct faults are indistinguishable they can be represented as one fault. These approaches greatly reduce computational cost. However, they depend on one single decomposition determined a priori. The approach of this paper is quite different: it dynamically constructs and modifies the decomposition as diagnosis proceeds and is complementary to these fixed approaches.

Sec. 5 demonstrates the application of tiered inference to production plant diagnosis. Consider a production system, where raw material is transported through a sequence of modules (known as an "itinerary") and modified to produce a product. At the end of an itinerary, one observes a good product or a damaged product. The product is damaged if any of the modules in the itinerary malfunctions. Furthermore, damage caused by a defective module cannot be repaired by subsequent modules. In this paradigm, diagnosis aims at isolating broken modules based on the itineraries and observed output. For this diagnosis problem, we analyze the tradeoff between computational cost and inference accuracy. While we use production plant diagnosis as an illustration, the ideas presented in this paper are more general and can be extended to other diagnosis problems.

2 Tiered inference

To mitigate the computational difficulty, we further advance our prior work in (Kuhn and de Kleer 2008) and propose the notion of *tiered inference*. The basic idea is to restrict posterior computation to a subset of hypotheses, and broaden the scope of inference only when necessary. In the tiered inference framework, we partition the overall hypothesis space into tiers, i.e.,

$$\mathcal{X} = \mathcal{X}_0 \cup \mathcal{X}_1 \cup \mathcal{X}_2 \cup \cdots \cup \mathcal{X}_M, \tag{3}$$

where each tier $\mathcal{X}_j$ is defined as the collection of hypotheses assuming a total of j faults in the system, i.e., hypotheses with cardinality j ($\sum_i x_i = j$). Once the system is observed to be malfunctioning, the need for diagnosis arises. Inference starts with the single-fault tier $\mathcal{X}_1$, assuming that the system has only one fault. At this tier, the inference only updates the posterior for the hypotheses in $\mathcal{X}_1$ and ignores all other hypotheses. This drastically reduces the computational complexity from $O(2^M)$ to $O(M)$. However, the single-fault assumption is an approximation, as the system can have multiple faults. When a conflict is detected, i.e., all the hypotheses in $\mathcal{X}_1$ conflict with the observation data, we escalate the inference to the next tier $\mathcal{X}_2$, assuming a

total of two faults in the system. The inference then updates all hypotheses in $\mathcal{X}_2$. The process repeats until observation data or the hypothesis space is exhausted.

Before diving into technical details, we first provide some intuition using an example. Figure 2 shows the computation structure in the tiered inference framework. The hypothesis space $\mathcal{X}$ is partitioned into non-overlapping tiers $\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_M$ as shown in Figure 2a. Figure 2b shows the computation in the tiered inference algorithm. Imagine a sequence of observations as follows:

1. The first batch of observations is used to update all hypotheses in $\mathcal{X}_1$, hence the computation is linear in $|\mathcal{X}_1|$. In Figure 2b, this is shown as vertical solid lines in first tier (the upper-left corner). The length of the lines symbolizes the amount of computation, in this case proportional to the size of $\mathcal{X}_1$.
2. The last observation of the first batch rules out all hypotheses in $\mathcal{X}_1$. In this case, we are forced to escalate to the double tier $\mathcal{X}_2$. The observations now need to be re-applied. This corresponds to the solid lines in the second tier. The computation is linear in $|\mathcal{X}_2|$.
3. The second batch of observations are applied to all hypotheses in $\mathcal{X}_2$. The computation is shown as the dashed lines in the second tier.
4. The last observation of the second batch further rules out all hypotheses in $\mathcal{X}_2$. Now we escalate to $\mathcal{X}_3$ and re-apply all the previous observations (solid and dashed lines in the third tier). As more observations are accumulated, the update computation (dotted lines in the figure) is restricted to $\mathcal{X}_3$.

In contrast, Figure 2c shows the computation where all observations are applied to all hypotheses. Notice that the total vertical lines are much shorter in Figure 2b than in Figure 2c. The computational savings are clear. The savings are primarily due to the fact that the higher tier hypotheses are not updated until necessary.

In this tiered inference framework, what is the price to pay in return for the inference computational savings? First bear in mind that this is an approximation — we have ignored the higher tiers when the lower tiers remains consistent with the observations. Therefore tiered inference loses optimality, for instance, the maximum a posterior (MAP) diagnosis is only optimal within the tiers that had been worked on. One can no longer claim optimality in the overall hypothesis space. Secondly, the tiered inference framework needs to store *all past observations*. In the case where the current tier is ruled out, the past observations will be re-applied to the new tier. This means the system should have enough memory. The comparison is as follows: If the computation is done sequentially each time a new observation is made, the memory storage requirement for updating the whole hypothesis space is 2^M — only the posterior probabilities need to be stored, the observation itself does not need to be stored. In contrast, the

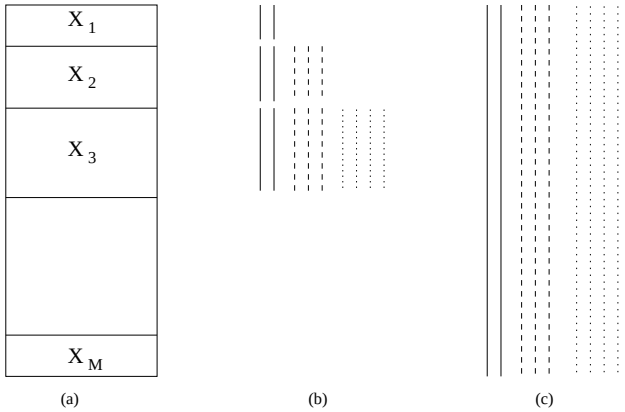

Figure 2: Computational structure: (a) partition hypothesis space into tiers, (b) computation in the tiered inference framework, (c) computation in the whole hypothesis space.

memory requirement for the tiered inference method is $|\mathcal{X}_j| + O(|observations|)$, i.e., we need to store the probability of hypotheses in the current tier, as well as all observations in the past. When the observation history is long, the memory requirement is high. In essence, the tiered inference framework reduces the burden on computation, but shifts the burden to memory storage. In practice one may be able to compress the observation history into some aggregated form.

It is important to characterize when this tiered inference framework is advantageous. In practical systems, most modules are likely to be good, and the total number of faults is likely to be small. In this case, the single-fault tier can be much more probable than the double-fault tier, and even more so than the triple-fault tier, and so on. Hence it makes sense to focus computational resources to the single-fault tier, and escalate to the higher tiers only when necessary. The higher tier hypotheses are safely ignored because they have minimal probability to start with. The computational savings are tremendous. On the other hand, a pathological case would be the situation where each module has a high (close to 1) probability of having fault. From the computational point of view, starting from the low cardinality tiers is less attractive, since the low cardinality hypotheses are likely to be ruled out by the observations, and the reduction in inference computation is less significant. Furthermore, as we shall see shortly, the tiered inference framework will incur an overhead cost of defining the next subset or tier of hypotheses to work on every time an existing tier is ruled out. This overhead cost will be high in this pathological case, making the tiered inference framework less attractive. On the flip side, this pathological case is rare.

3 Partition into single-fault subsystems

Diagnosing a single-fault is computationally efficient. If a M-module system is assumed or known to have a single-fault, we only need compare M hypotheses, rather than the 2^M hypotheses in the multi-fault case. Given that single-fault inference is computationally efficient, it would be nice to apply this technique whenever applicable. This motivates us to find single-fault subsystems although the overall system can have multiple faults.

The tiered inference idea in the previous section suggests that we can use single-fault diagnosis in the first tier $\mathcal{X}_1$ until data conflict arises. Figure 3 shows a simple example system with only 4 modules ($ABCD$). Figure 3a arranges the hypotheses based on their cardinality. This defines the tiers $\mathcal{X}_0$, $\mathcal{X}_1$, $\mathcal{X}_2$, and so on. In the tiered inference framework, we start from $\mathcal{X}_0$ and $\mathcal{X}_1$. When data suggests that the system ($ABCD$) has more than one faults, the tiered inference escalates to the double-fault tier, $\mathcal{X}_2 \triangleq \{\mathbf{x} : \sum_i x_i = 2\}$, as shown in Figure 3b. At this point, we know that the overall system ($ABCD$) has at least two faults, but it is possible that subsystems, for instance, (AB) and (CD) each has a single fault. In this case, we can still apply single-fault diagnosis to subsystems (AB) and CD separately to isolate the faults. The computation is still efficient. With this partition, the update is restricted to hypotheses into the subset $\mathcal{X}^t = \{\mathbf{x} \mid x_A + x_B \leq 1, x_C + x_D \leq 1\}$, shown as the hypotheses marked with check-marks in the top box in Figure 3c. The computation is restricted to $\mathcal{X}^t$, hence fast.

The question now is to seek a good partitioning such that the partitioned subsystems are most likely to have single fault. Formally, the partitioning problem is as follows: given an overall system S containing modules, the partitioning divides S into two groups S_1 and S_2 such that $S_1 \cup S_2 = S$ and $S_1 \cap S_2 = \emptyset$. For instance, in the example in Figure 3, $S_1 = (AB)$ and $S_2 = (CD)$ is a valid partition. Note that this partitioning is not unique: we can partition ($ABCD$) into $\{(AB), (CD)\}$ (top box in the figure), or $\{(AD), (BC)\}$ (second box in the figure)[1] or other combinations. The next section addresses the question of which partition to use. The basic idea is to examine the correlation between system components to find those subsets which collectively contain only a single fault with maximum probability.

Given a subsystem partition and the corresponding subset of hypotheses $\mathcal{X}^t$ assuming at most a single fault within each subsystem, the algorithm restricts the posterior updates to the subset, until the observation data conflicts with $\mathcal{X}^t$. In this case, we backtrack to the existing tier $\mathcal{X}_2$ and find a more suitable partition. When the whole tier $\mathcal{X}_2$ is ruled out by observation, we escalate to the third tier $\mathcal{X}_3$ (the collection of hypotheses with 3 fault modules) and partition the overall system into three subsystems, each of which hopefully contains a single fault. The whole process repeats as more observations are made.

[1] We use the bracket to denote a group within which there is believed to be only single-fault, and the curly bracket for a collection of groups.

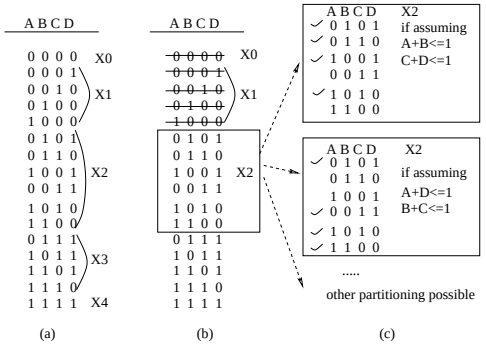

Figure 3: Example of tiered inference: (a) hypothesis space and tiers; (b) escalating to tier $\mathcal{X}_2$; (c) partition $\mathcal{X}_2$ into two groups, each with (at most) a single fault: top box — partition into $\{(AB), (CD)\}$; second box — partition into $\{(AD), (BC)\}$. Other partition are also possible.

4 How to partition

4.1 Criterion for partitioning

As mentioned in the previous section, when the single-fault assumption fails, we escalate to $\mathcal{X}_2$ and assume that the overall system has two faults. We partition the M-module system into two subsystems, or two groups, within which there is likely to be at most one fault.

There are many ways of partitioning a system into two groups. For example, ($ABCD$) can be partitioned into $C_4^1 + C_4^2/2 = 7$ ways. Which one is more preferable? What optimality criteria should we use? The intuition is clear: we would like to make sure that the single-fault assumption for each subsystem is true with maximal probability.

Criterion: We favor the partition (of the module set) which captures maximal probability mass, i.e., maximizing the probability $\sum_{\mathbf{x} \in \mathcal{X}^t} p(\mathbf{x})$.

For instance, in Figure 3, partitioning into subsystems $\{(AB), (CD)\}$, shown as the top block on the right hand side, captures hypotheses $\{0101, 0110, 1001, 1010\}$. There are two hypotheses $\{0011, 1100\}$ that violates the single-fault assumption in (CD) and (AB) respectively. If the probabilities $p(0011)$ and $p(1100)$ are small, this means (AB) and (CD) are likely to have single-fault, and the partition is advantageous. On the other hand, if $p(0011)$ and $p(1100)$ are big, this mean the single-fault subsystem assumption is questionable. To compare the two partitions $\{(AB), (CD)\}$ and $\{(AC), (BD)\}$, we only need to compare the probability mass of missed hypotheses, in this case, $p(0011) + p(1100)$ and $p(0110) + p(1001)$.

The partition with a lower probability mass is more favorable.

4.2 A partitioning algorithm

Now with the optimality criterion, how should we design the partitioning algorithm? The straightforward solution is to compare all partitions and see which partition captures the largest probability sum, but this is too expensive with complexity 2^M. Can we find a partitioning which is good (of course suboptimal) with much less computation time? We first discuss the case of partitioning into two groups.

> **Intuition:** For a group of modules to have a single fault, i.e., $\sum_{i \in P} x_i = 1$, the x_i's would have to be negatively correlated.

In other words, when one member x_i increases, there must be another x_j which decreases in order to maintain the constant sum. This means we should look for modules with significant negative correlation and group them into a group. In contrast, if two members are positively correlated, i.e., when one increases/decreases, the other one increases/decreases too, then these two modules should not be grouped into the same group.

Using this heuristics we propose an algorithm, which examines the correlation coefficient between modules. The correlation coefficient is defined as

$$\begin{aligned} \rho(i,j) &\triangleq \frac{Cov(x_i, x_j)}{\sigma_i \sigma_j} \\ &= \frac{E\left[(x_i - \mu_i)(x_j - \mu_j)\right]}{\sigma_i \sigma_j} \end{aligned} \tag{4}$$

For any two modules i and j, x_i and x_j are the indicators of their respective health (0 if the module is good, and 1 if the module is bad), μ_i and μ_j are the respective mean of x_i and x_j, and σ_i and σ_j are their respective standard deviations. The correlation coefficient $\rho(i,j)$ measures the dependency between x_i and x_j. It has the following properties: (a) $-1 \le \rho \le 1$; (b) the sign of ρ shows whether the two random variables are positively or negatively correlated; (c) $\rho = 1$ if $x_i = x_j$, and $\rho = -1$ if $x_i = -x_j$; (d) symmetry: $\rho(i,j) = \rho(j,i)$. Given a set of hypotheses $\{\mathbf{x}\}$ and their respective probability values, one can easily compute the mean $\{\mu_i\}_{i=1,\ldots,M}$, the standard deviation $\{\sigma_i\}$, the covariance matrix $\{Cov(x_i, x_j)\}_{i,j=1,\ldots,M}$, and the correlation coefficient $\rho(i,j)$ for any i and j. The computational complexity is linear in the number of hypotheses.

The algorithm is the following:

1. From the hypotheses and their respective probabilities, evaluate the correlation coefficient $\rho(i,j)$ for any (i,j). The result is a correlation coefficient matrix of size $M \times M$.
2. Find the two group seeds i_1 and i_2 as the module which have the highest correlation $E(x_i^2)$ values. This indicates that these two modules are more likely to have a fault than the others. In the case of a tie, we select seeds randomly. The two groups "grow" around the seeds. Previously we have used a random selection scheme: randomly select the first seed i_1, and then find the second group seed i_2 as the module which has the highest correlation with i_1. Since these two are positively correlated, they should not be in the same group. The max-correlation scheme works best in our simulations.
3. For any remaining module j, compare the correlation coefficients $\rho(i_1, j)$ and $\rho(i_2, j)$. The module is assigned to group 1 if $\rho(i_1, j) < \rho(i_2, j)$ and to group 2 if otherwise.

Computational complexity — The computation is primarily on the computation of $\{\rho(i,j)\}$. The complexity if $O(M^2 \cdot |\#\text{ of hypotheses}|)$ — there are M^2 correlation coefficients, and computing each need to go through all hypotheses in the current tier. In contrast, the "oracle" scheme of comparing all partitioning combinations has complexity $O(2^M \cdot |\#\text{ of hypotheses}|)$.

Performance — Despite its simplicity, this greedy algorithm works well. In our simulation, we repeated for a large number (100) of random simulations, and compared this partitioning scheme against the enumeration of 2^M possible partitions. Our partition selection scheme has the following performance:

- Against the missing probability metric: our partition selection method is at about the 85% percentile among all 2^M partitions, i.e., around 15% partitions are better than our solution, and 85% are worse. But the computational complexity is much less.
- Compared to the "oracle" — the partition with smallest missing probability, our partition scheme produces a slightly larger missing probability, on average 3–5% larger.

Example — Consider a 5-module production system $(ABCDE)$. The observations are as follows: (1) observing a fault with itinerary $(ABCDE)$; (2) observing a fault with itinerary (ABC); (3) observing a fault with (DE). At this point, the single fault assumptions are eliminated. We assume each module is defective with a prior probability $r = 0.1$. We further assume all faults are persistent. In this case, the covariance coefficient matrix is:

$$\rho = \begin{pmatrix} 1 & -0.5 & -0.5 & 0 & 0 \\ -0.5 & 1 & -0.5 & 0 & 0 \\ -0.5 & -0.5 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & -1 & 1 \end{pmatrix} \tag{5}$$

The partitioning algorithm selects B and D as group seeds and partitions modules into two subsystems (ABC) and (DE), which agrees with our intuition.

A similar problem is optimal number partitioning (Korf 1995), which partitions a set of integer numbers into two groups with equal sums. However, there is a fundamental difference: the optimal number partitioning is deterministic, while our partitioning problem

is inherently statistical and must work with uncertainties. As a result, the algorithms for the two problems are quite different.

4.3 Preparing probability distribution for partitioning

The algorithm above requires the computation of correlation coefficients $\{\rho(i,j)\}_{i,j=1,2,\ldots,M}$. They are computed based on a set of hypotheses and their respective probability values. Should this hypothesis set be the entire hypothesis space ($\mathcal{X}$, size 2^M)? or a smaller subset? We argue that it may be sufficient to compute the distribution for a subset. For instance, if the first tier (the single fault hypotheses tier $\mathcal{X}_1$) is ruled out, and we must escalate to double faults, we only need to examine the double fault hypothesis tier $\mathcal{X}_2$, since other hypotheses are out of the representation of two-group partition anyway. Therefore the other hypotheses will not be covered by the partitioning. In our tiered inference framework, we use tier $\mathcal{X}_2$ for partitioning into two groups. Likewise, if $\mathcal{X}_2$ is ruled out by observations, we escalate to the triple-fault tier $\mathcal{X}_3$, and partition the M-module system into three groups. The partitioning is computed based on the probability values of all hypotheses in $\mathcal{X}_3$.

The algorithm described above can be modified to partitioning components into any number of groups. The extension is straight-forward: we just select more group seeds in Step 2, and let the seeds grow into groups.

5 Implementation and simulation

As an example to illustrate the advantages and drawbacks of the tiered inference approach, we consider diagnosis of a production plant. Assume that modules are independent, and each module is defective with a known prior probability r. All faults are intermittent, i.e., a defective module damages any product passing it with a known probability q, known as the intermittency probability. In practice, each module may have its own r and q, different from the others. In our implementation, for simplicity, we assume that all modules share the same r and q value.

Mathematically, we have the prior probability

$$p(\mathbf{x}) = \left(r^{\sum_i x_i}\right) \cdot \left((1-r)^{M-\sum_i x_i}\right).$$

Given an itinerary w, the likelihood of observing an output o (0 for good, and 1 for damaged) is

$$p(o|\mathbf{x}) = \begin{cases} (1-q)^{k(w,\mathbf{x})} & \text{if } o=0 \\ 1-(1-q)^{k(w,\mathbf{x})} & \text{if } o=1 \end{cases}$$

Here the exponent $k(w,\mathbf{x})$ is the number of defective modules involved in the production itinerary w given the hypothesis $\mathbf{x}$. This is actually quite intuitive: a product is undamaged only when none of the defective modules malfunctions, hence the probability is the module-wise good probability $(1-q)$ raised to the power $k(w,\mathbf{x})$.

Now with prior and likelihood probabilities specified, we perform Bayesian updates (Equation 1). Two diagnosis schemes are compared: (a) a baseline scheme applying all observations sequentially to update the posterior belief $p(\mathbf{x}|o)$ for all $\mathbf{x} \in \mathcal{X}$ that has not been ruled out by previous observation data; and (b) the tiered inference scheme described in Secs. 2– 4. To evaluate the performance, we simulate 300 random trials, each with an observation sequence of 400 randomly generated production itineraries and corresponding outputs. Performance are assessed based on cost and accuracy:

- Computational cost: for the baseline scheme, computational cost is measured as the accumulative number of posterior updates, i.e., how many times (Equation 1) is executed. For tiered inference, the cost is the sum of two parts: (i) the inference cost, i.e., the number of posterior updates, and (ii) the overhead cost of partitioning modules into subsystems, measured as the number of hypotheses sieved through to compute the correlation coefficient (Equation 4). Table 1 reports the two terms, separated by a ";" in the third column.
- Diagnosis accuracy, measured as the total number of bits that $\mathbf{x}_{MAP}$ differ from the ground truth. Ideally, if $\mathbf{x}_{MAP}$ recovers the ground truth, this term should be 0. However, this is often not achieved, even in the baseline inference scheme. This is due to the fact that the observations may not be sufficient, for instance, if some defective modules are never used in production, and/or the faults are intermittent, hence the defects are never observed.

Table 1 reports the results for a 10-module production system, averaged over 300 random trials. Each row corresponds to a value of r, ranging from 0.05 to 0.9. Small r implies a healthy system, while $r = 0.9$ corresponding to an extremely shaky system where all modules are likely to fail. We use the extremes to provide insights. Note the following:

(1) The computational cost saving using the tiered inference scheme is significant. For instance, with $r = 0.05$, the tiered inference scheme has a computation cost less than 1% of the baseline scheme. With $r = 0.9$, the tiered inference computation is around 10% of the baseline computation.

(2) The baseline scheme is on average more accurate than the tiered inference. This is expected, since the tiered inference is an approximation.

(3) Tiered inference is most advantageous when r is small. The inference accuracy is almost as good as the baseline scheme for $r \leq 0.2$, and the computation cost is 1–2 magnitudes order lower. This shows the benefit of tiered inference. The good performance is not surprising, as a system with small r is what tiered inference was originally designed for.

(4) As r increases, tiered inference incurs a increasingly heavy partitioning overhead cost (second number in the third column). This is due to the fact that the system has more defective modules, and the single-

	Computation cost		Diagnosis accuracy	
	baseline	tiered	baseline	tiered
$r = 0.05$	285779.7	2268.5; 63.3	0.03	0.03
$r = 0.1$	265003.1	2051.9; 448.3	0.17	0.13
$r = 0.2$	236468.3	2705.2; 1435.7	0.47	0.59
$r = 0.5$	175757.8	6293.7; 4610.0	1.51	2.36
$r = 0.9$	141973.8	7875.2; 6470.0	1.16	5.07

Table 1: Tradeoff between computational cost and diagnosis accuracy. This table is generated assuming the intermittency probability of $q = 0.1$. The second column reports computation cost of the baseline scheme measured as the number of hypotheses updated; the third column reports the computation cost for tiered inference for Bayesian update and partitioning overhead, and the last two columns report diagnosis accuracy of the two schemes, measured as the number of bits that MAP estimate differs from the ground truth.

fault assumption within subsystems is often ruled out by observation data. In this case, the partitioning operations are frequently repeated. The overhead cost makes computational savings less dramatic. Furthermore, tiered inference becomes less accurate. For instance, in the last row ($r = 0.9$), the tiered inference diagnosis has roughly 5 bits flipped. It misses to detect 5 defective modules. In comparison, the baseline has 1.16 bits flipped on average. Note that this is due to their different strategies: the baseline scheme seeks exact inference and optimal diagnosis, while tiered inference favors low-cardinality diagnosis. Tiered inference stays at lower tiers as long as the lower tiers can explain the data. This is similar to a minimal diagnosis: the minimal candidate set can be quite different from the underlying ground truth, especially when the faults are intermittent and the number of observations are limited.

6 Conclusion

This paper has presented a new framework for efficiently computing multiple fault diagnoses. This framework introduces the generic notion of tiered inference which focuses search and inference on the set of hypotheses most likely to contain the fault(s). Past approaches which focus on most probable, subset-minimal, or minimum cardinality approaches are all instances of the more general tiered approach. In addition, this paper introduced the notion of partitioning the modules such that efficient, linear, single fault inference can be used (and never requires the usual multiple-fault inference scheme). By performing single fault diagnosis on each partition, the potential computational inference on each partition is avoided. For smaller cardinality diagnoses, we believe the inference saving outweighs the cost of computing partitions (including recomputing partitions when they are discovered to be unsuccessful).

References

Berger, J. O. 1995. *Statistical Decision Theory and Bayesian Analysis*. Springer Verlay, New York.

de Kleer, J., and Williams, B. C. 1987. Diagnosing multiple faults. *Artificial Intelligence* (32):97–130.

Korf, R. 1995. Optimal number partitioning. Technical report, also available at ftp://ftp.cs.ucla.edu/tech-report/1995-reports/950062.ps.Z.

Kuhn, L., and de Kleer, J. 2008. An integrated approach to qualitative model-based diagnosis. In *Qualitative Reasoning Workshop (QR 2008)*.

Pravan, G. 2001. Hierarchical model-based diagnosis. In *Proc. International Workshop on Principles of Diagnosis (DX)*.

Reiter, R. 1987. A theory of diagnosis from first principles. *Artificial Intelligence* 32(1):57–96.

Siddiqi, S., and Huang, J. 2007. Hierarchical diagnosis of multiple faults. In *Proceedings of IJCAI*.

Srinivas, S. 1994. A probabilistic approach to hierarchical model-based diagnosis. In *Proc. Conference on Uncertainty in AI (UAI)*, 538–545.

Thiebuax, S.; Cordier, M.; Jehl, O.; and Krivine, J. 1996. Supply restoration in power distribution systems – a case study in integrating model-based diagnosis and repair planning. In *Prof.8th International Workshop on Principles of Diagnosis (DX)*.