Efficient Differential Diagnosis using Cost-aware Active Testing

Emile van Gerwen[1], Leonardo Barbini[2], Michael Borth[3], and Robert Passmann[4]

[1,2,4]*TNO-ESI, Eindhoven, Noord-Brabant, 5656 AE, The Netherlands*

emile.vangerwen@tno.nl
leonardo.barbini@tno.nl
robert.passmann@tno.nl

[3]*TNO, Helmond, Noord-Brabant, 5708 JZ, The Netherlands*

michael.borth@tno.nl

ABSTRACT

The diagnosis of complex systems benefits greatly from a differential, multistep approach that narrows down the list of possible conditions or failures that share the same observable effects to a single root cause. We provide a suitable and practically applicable methodology for this. In extension to existing work, it covers all types of diagnostic actions, i.e., the observation of system properties, active testing, and system interventions like providing a dedicated diagnostic input or forcing the system into discriminating states, but also the replacement of components. Combining all these possible steps into one probabilistic and causal reasoning framework, we I) stepwise generate the diagnostic model systematically to correctly cover the interplay of observations and diagnostic interventions, and II) provide decision support based on counterfactuals for the selection of the next diagnostic step, countering the vast number of possible actions that arise in machine diagnostic processes, considering costs of actions to minimize the expected overall cost of the diagnosis. We developed and successfully tried our methodology for diagnosing cyber-physical systems in the high-tech industry, but we found that it supports more processes, such as computing intervention actions for autonomous robots.

1. INTRODUCTION

Machine Diagnosis, understood as the identification of the nature and cause of a certain and unwanted phenomenon within a technical system by means of at least semi-automated analytics, is key in after-sales processes of many industries. It allows systems to partially self-diagnose, thus offering users of production equipment reduced downtimes either by enabling them to fix issues themselves or by ensuring that service personnel come in with the right parts for potential hardware failures. Further, it tunes maintenance towards minimal costs and counters the shortage of experienced service engineers by raising their efficiency and by enabling less experienced personnel to perform the tasks.

One crucial design decision in the realization of a system for machine diagnosis is which information it processes. Failure effects and measurements of key performance indicators are typical candidates that can be observed and processed, but experienced service engineers consider more: they use active testing, i.e., they set system states or provide the system with tailored inputs to discriminate between root causes but also undertake costly investigative part replacements.

We extend previous work for machine diagnosis (Barbini et al., 2020) by introducing automated reasoning for differential diagnosis with active testing and other interventions within a unified framework. For this, we lay out our methodology with causal Bayesian Networks in Section 2 and provide an example in Section 3 that shows the power of interventions in machine diagnosis. In Section 4, we introduce decision support for selecting the best action, extending our previous work (van Gerwen et al., 2023) with dynamic calculations on diagnostic costs. Section 5 illustrates two of our use cases, and Section 6 concludes with future research topics.

2. DIFFERENTIAL MACHINE DIAGNOSIS

2.1. Diagnosis using Causal Belief Networks

Bayesian Belief Networks, also known as Bayes Nets (BN), are probabilistic graphical models that were shown to deliver excellent diagnostic capabilities (Heckerman, 1995, among other works). Introduced by Pearl (1986), they offer probabilistic reasoning for taking an observable event and inferring the likelihood that any one of several possible known causes was the contributing factor. In this, they form an efficient solution to compute the marginal distribution of the known causes, i.e., of a hypothesis *H*, given observations or evidence *e*, following Bayes' Theorem:

https://doi.org/10.36001/IJPHM.2024.v15i3.3849

$$P(H|e) = \frac{P(e|H) * P(H)}{P(e)}$$

with *P(H)*, *P(e)* the probability of observing *H* and *e* (the priors or marginal distributions), and *P(H|e)*, *P(e|H)* the conditional probability of *H* and *e*, given the other one. Defined as directed acyclic graphs (DAGs) whose nodes represent variables in the Bayesian sense and whose edges represent conditional dependencies, BNs encode the joint probability distribution over all their variables. Their advantage for the purpose of diagnosis stems in part from the versatility that those variables may be observable, latent, and even unknown parameters or hypotheses, and that, with the joint probability distribution, all arbitrary combinations of variables may be considered. With this, BNs consider all causes that are possible given the observations and rank them according to their likelihood that depends on observations.

As Bayes nets are one of the few inference techniques that combine knowledge- and data-driven modeling (Jensen & Nielsen, 2007), there are many options to build or generate BNs. Our approach for the construction of BNs for diagnosis is based on modeling the causal knowledge on the system decomposition, deployment, and functional behavior and the use of generative techniques to compose the needed BN from pre-build network fragments and rule-based elements such that it conforms to that knowledge. Introduced by Borth & von Hasseln (2002), this process translates system knowledge that inherently adheres to the Markov condition directly to Pearl's interpretation of structural causal models, offering automated network generation that we seek for industrial use based on (Pfeffer et al., 1999), thus motivating our choice for Bayesian networks. We detailed and extended this approach in (Barbini & Borth, 2019) and (Barbini et al., 2020), but, so far, limited ourselves to diagnostic processes that only use a set of simultaneous observations or a sequence of observations – but not a mixed sequence of observations and interventions. In essence, we (and others, compare, e.g., Ricks & Mengshoel, 2009) generated diagnostic systems that use the full power of causal Bayesian Belief Networks which mirror a system as it is, but did not utilize the diagnostic prowess of interventions to the system. A major reason for this restriction was that interventions alter the system's causality – and neither modeling techniques nor inference calculi were ready to handle this until quite recently.

This changed with Pearl's *do(x)* operator (2009) that represents interventions and allows to correctly predict effects of such deliberate actions in causal networks. *do(x)* operator interventions have a different meaning and diagnostic power than statistical associations (Pearl & Mackenzie, 2018): *P(Y|x) > P(Y)* simply states that observing *x* raises the probability of *Y*, which might have other reasons including a common cause, while *P(Y|do(x))* describes the situation after performing the action *x* that affects *Y* and eliminates the effects of other confounding factors. Within diagnosis, this translates, e.g., to the difference between observing the availability of power at a connector – with the conclusion that the functional chain to provide that power is intact – and experimentally providing power at that point, invalidating all assumptions about that chain while offering deductions from the effects observed due to the intervention.

2.2. Differential Causal Reasoning with Bayes Nets

Within the probabilistic graphical model, the distinction of observing *x* and *do(x)* is paramount, as the first provides evidence for the variable *x*, i.e., setting the probability of the observed state to 1, leaving the rest of the network as it was, while the second also changes the network structure to reflect the new causal flow. A differential diagnosis using a mixed sequence of observations and interventions therefore requires a new technique of stepwise inference and network augmentation. Figure 1 depicts our methodology for this.

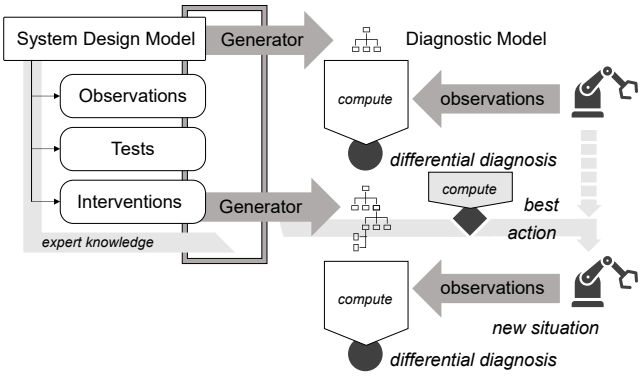


Figure 1. Differential Machine Diagnosis

As the graph shows, the diagnostic model is generated from the system design, with augmentations to the diagnostic model following subsequently for conducted actions that change the diagnosis and, in the case of interventions, the system (Section 3). The observations from the system itself allow us to compute the differential diagnosis at each step.

Using this process and given the encoded expert knowledge, we generate diagnostic models that I) accurately represent a sequence of diagnostic tests of all kinds, i.e., the observation of additional properties, the setting of a different input, system interventions, and the replacement of components, and II) correctly carry the information between steps.

The resulting differential diagnosis covers single fault as well as multi-fault scenarios. Intermittent faults, however, cannot

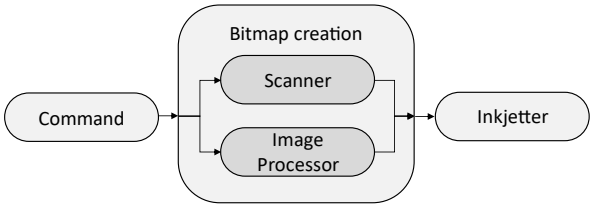


Figure 2. Schematic representation of an all-in-one printer

be handled directly in this way (or any comparable approach), as the absence of a fault effect observation potentially leads to the diagnostic network ruling out the respective root-cause.

3. INTERVENTIONS

We use an idealized version of a printer as an example to illustrate our approach to handle interventions in differential diagnosis. Figure 2 shows a schema of the idealized printer.

Based on a given *Command*, either *print* or *copy*, the printer should create a *Bitmap* and produce a printed sheet with the *Inkjetter*. For the *Command copy*, the printer needs only the *Scanner* working to produce a *Bitmap*. Similarly, for the *Command print*, it needs only a working *Image Processor* to produce a *Bitmap.* The BN to diagnose the printer is shown in Figure 3, using [Bayes Server]. Each system component is modelled with inputs, outputs, and health node. The system is created by considering the outputs of a component as the inputs of the following component. For details on how to generate this BN, we refer to (Barbini et al., 2020).

3.1. System and Component Health

In the BN figures below, nodes with a green identifier in the upper right corner represent the hidden health variable of components. In this example, we assume that the *Command* never fails, so it does not have a health node. The *Scanner* is *broken* with prior probability 5%, the *Image Processor* with 1% and the *Inkjetter* with 10% respectively. The other nodes in the BN are not hidden, i.e., are nodes for which the state can be measured.

For our diagnostic task, we use the BN to infer the probability distribution of the health nodes, given evidence in the other nodes. An example of how to perform a diagnosis with the BN is shown in Figure 3. We suppose that the *Command* is set to *copy*, but we do not obtain a *Printed Sheet.* The diagnosis, as the inferred probabilities on the health nodes, is very uncertain and the goal of our differential diagnosis is to perform diagnostic tests to reduce such uncertainty.

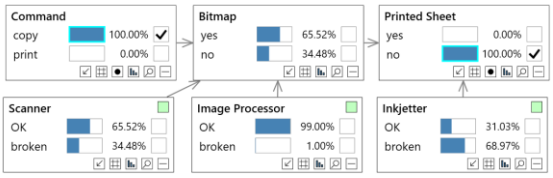


Figure 3. BN of all-in-one printer. Copying fails.

3.2. Diagnostics Tests

There are four different types of diagnostic tests. Each type comes with specific modeling and BN interactions.

Type I: observe additional properties

The first type of diagnostic test, widely covered by existing work on BN for diagnosis, consists of adding additional evidence to one of the observable nodes. This is shown in Figure 4, for the observation of an existing *Bitmap*. This test is executed by probing the printer to collect the additional observation on the presence of the *Bitmap*.

Figure 4. Type I: Observe additional properties

Type II: set a different input

A second type of diagnostic test consists of setting a different input to the system. This is a fundamentally different type of test than the previous one. Firstly, we are not only observing, but we are intervening in the system, by doing the action of changing an input. Secondly, the result of the action could potentially lead to a different observation than in the absence of said action, making it counterfactual.

Our approach is to augment the BN when doing a test comprising an intervention. This augmentation procedure has two steps. Firstly, we duplicate all the nodes in the BN except for the *health* nodes and the *input* nodes which are not set to a different value (with *input* nodes referring to BN nodes that have no parents and are not of type *health*). Secondly, we connect the duplicated nodes as children of the original *health* nodes. This approach – and not, as is sometimes assumed, just updating the priors using the result of the inference – delivers the correct inference model, as shown in (Balke & Pearl, 1994). Notice that with this augmentation procedure there is no need to specify new conditional probability tables, as they are the same as in the original, not augmented BN.

Figure 5 shows an example of this augmentation procedure. The starting point is the diagnosis BN in Figure 3. The duplicated nodes have a *1* at the end of their names, e.g., *Command1*. We now observe a *Printed Sheet1* for the printer input *Command1 print*. As the augmented BN's reasoning also includes the evidence coming from the situation when copying failed, it infers that the only possible diagnostic solution is that the *Scanner* is *broken* and the *Image Processor* and *InkJetter* are *OK*.

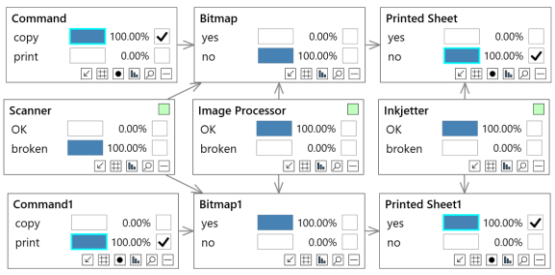


Figure 5. Type II: set a different input

Type III: replace a component

The third type of diagnostic test is the replacement of a component. This is also an active type of test, and thus of a similar nature as the previous one. When doing differential diagnosis with this type of test we also need to augment the BN, but – differently than with Type II tests – we duplicate its health node when we replace a component.

Figure 6 shows this procedure on the idealized printer for the case that we replace the *InkJetter* after copying fails. For the replaced component we assumed that *InkJetter1* functions (this is not necessary and might be relaxed). The *input* node *Command* is not set to a different value when the printer has a new *Inkjetter*, and is thus not duplicated, making it a common parent for both the *Bitmap* and *Bitmap1* node.

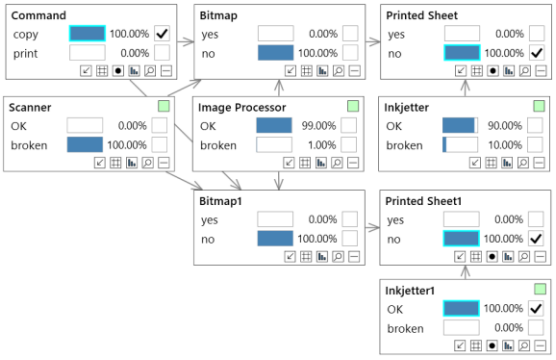


Figure 6. Type III: replace a component

The observation related to the action of replacing the *InkJetter* is that there is still not a *PrintedSeet1*. The BN infers that a *broken Scanner* explains the situation, allowing even a switch back to the original *InkJetter*, as the diagnosis established that the failure had to progress via the *Bitmap*, a statement that could not have derived from another probe.

Type IV: intervene within the system

The fourth type of test consists of forcing physical quantities to a given value. In practical situations this corresponds to performing an action in the system, like connecting an external power supply to a part, and then observing the results in other parts, for example whether these are now correctly functioning. This test type also entails an intervention on the system and modeling it requires both an augmentation of the BN and the *do(x)* operator introduced above.

Figure 7 shows an example of type IV testing: After copying fails, we try to print, which is a Type II test, but then we still do not get a *Printed Sheet1*. At this point, we intervene in the system and manually provide a *Bitmap2* to the *InkJetter*. The light gray arrows pointing into *Bitmap2* node indicate the obstruction of information flow.

The BN is consequently augmented with the duplication procedure described above, but in this type of test, we insert evidence with the *do(x)* operator on the duplicated node on which we force its physical quantity to a given value. In the case of Figure 7, this is the *Bitmap2* node, and the use of *do(x)* to set the evidence is shown with a red check mark.

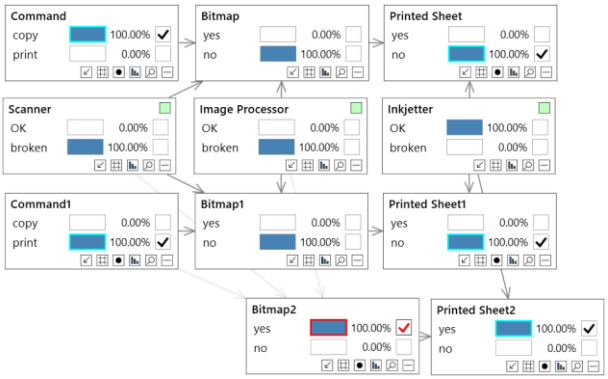


Figure 7. Type IV: intervene within the system

In our example, we finally observe a *Printed Sheet2* after providing *Bitmap2*. The augmented BN infers that the only possible solution to the diagnostic problem is that both the *Scanner* and the *Image Processor* are *broken* and the *InkJetter* is *OK*.[1]

4. DECISION SUPPORT

A major part of our system for machine diagnosis is to recommend the next best action to service engineers, as this increases their efficiency and reduces system downtime. To do so, we assess the value of possible diagnostic tests, i.e., we determine how much we would learn from them using counterfactual <what-if> reasoning. A test can be of any of the types outlined in the previous section. The result of each <what-if> scenario is the information gain or equivalently the reduction of uncertainty, for which we use an entropy measure (Oladyshkin & Nowak, 2019).

Ideally, we would consider all possible test sequences and pick the one that is expected to lead to the diagnostic solution with minimal effort.

However, considering that for large systems there are hundreds of possible tests at any point, this approach seems infeasible. Instead, encouraged by (de Kleer et al., 1992) stating that “one step lookahead is pretty good”, we take the myopic approach in which we look only at the currently available tests, and pick the test based on the expected entropy within the BN after doing that single test.

[1] Without the *do(x)* operator, this evidence on *Printed Sheet2* could not be set. It would have a null probability of finding as we tried both printing and copying and both failed. The *do(x)* operator deletes the dependence of *Bitmap2* on its parent nodes and as a result one can have a *Bitmap2* even if both the *Scanner* and the *Image Processor* are *broken*.

4.1. Entropy for Measuring Diagnostic Impact

Intuitively, the entropy over the component health states is a measure of the uncertainty we have about the health of these components. The diagnostic process is then doing tests to reduce the entropy as much as possible.

If we denote the set of health nodes in the Bayesian network with Ω, conditional entropy in model M with $E_M(\cdot \mid \cdot)$, and $T_i = (A_i, O_i)$ a possible test that comprises of action A_i and observation O_i, we calculate the recommended test T^* as

$$T^* = \underset{T_i}{\operatorname{argmax}}(E_M(\Omega) - E_{M \leftarrow A_i}(\Omega \mid O_i)), \quad (1)$$

where M is the model representing the current situation and $M \leftarrow A_i$ the hypothetical situation if we would perform action A_i on model M as described in Section 3. Notice that in the same T_i the action and the observation can be performed in different system's components. Calculating T^* thus requires generating networks for every action A_i.

Once the recommended test is executed, we generate the corresponding network, add the evidence that reflects the outcome of the test, and calculate the posterior probabilities of the components' health, thereby presenting the new failure hypotheses. This procedure is repeated until a solution to the diagnostic problem is found.

4.2. Dynamic and Fixed Costs of Tests

Equation (1) does not consider the costs of various tests, such as time spent and parts expenses. In industrial practice, these must be considered to achieve low total costs of ownership.

It is crucial to note that such costs are not static but depend dynamically on the system's current state. For example, the time of replacing a component (Type III above) depends not only on the specific component but also on the necessary work to make the system ready for the replacement. This work is dynamic as, for example, some cover of the machine might already have been removed for a previous test.[2]

Including dynamic costs is not only important for a proper estimate of the effort required for the next test but also crucially influences the order in which tests are recommended: a test becomes cheaper when its dynamic costs are lowered due to a previous test having been performed. In other words, if a test has complicated requirements that are currently met, it may be sensible to perform it even though another test is more insightful, i.e. it reduces the entropy more strongly.

We first describe the abstract mathematical framework for incorporating dynamic costs, and then apply it to a concrete example for illustration. In what follows, we assume that costs are non-negative numbers (i.e. one dimensional; generalizing to different types of costs is straightforward).

A *system state* is a function $s: \{1, \dots, n\} \to \{0, 1\}$. We will use the notation $s(i)$ to denote the value of a so-called *system condition* i in state s. In other words, a system state is an instantiated configuration of all system conditions. Semantically, $s(i) = 1$ corresponds to condition i being true in system state s, and $s(i) = 0$ to condition i being false in system state s.

To aggregate the dynamic cost of a test, we need to know the costs of changing the system conditions. To do so, we introduce a *dynamic cost function*, i.e. a map

$$c: \{1, \dots, n\} \times \{0,1\} \to \mathbb{R}^{\geq 0}, \quad (2)$$

where $c(i, 0)$ is the cost of setting condition i to 0, and $c(i, 1)$ is the cost of setting condition i to 1.

To every diagnostic test T_i, we assign a (potentially empty) set C_i of tuples (j, b), where $1 \leq j \leq n$, and $b \in \{0, 1\}$. The intended meaning is that $(j, 0) \in C_i$ if and only if test i requires condition j to be false, and $(j, 1) \in C_i$ if and only if test i requires condition j to be true. If a condition j does not occur in any tuple in C_i, then the intended meaning is that test i depends neither on c_j being true nor on c_j being false.[3]

Finally, given a system state s, a dynamic cost function c, and a set of conditions C_i for a diagnostic test T_i, we can compute its dynamic cost as follows:

$$c^s_{dyn}(T_i) = \sum_{(j,k) \in C_i \text{ with } s(j) \neq k} c(j, b) \quad (3)$$

The total cost of performing a diagnostic test consists of the sum of its fixed costs, denoted by $c_{\text{fix}}(T_i)$, as well as its dynamic costs:

$$c^s_{total}(T_i) = c_{fix}(T_i) + c^s_{dyn}(T_i) \quad (4)$$

We left the system state implicit in the previous equation; if necessary, it may be denoted by adding a superscript s to any of the cost functions that depend on it.

When a diagnostic test T_i is performed, the system state s is updated to s_{new} according to the conditions required by the diagnostic test:

$$s_{new}(j) = \begin{cases} k, & \text{if } (j, k) \text{ in } C_i, \\ s(j), & \text{otherwise.} \end{cases} \quad (5)$$

[2] We consider *dynamic* costs to be those that change during the diagnostic procedure. We do not consider costs that depend on external factors rather than the system's state, for example, the time to replace may depend on the engineer's level of proficiency, or the availability of parts in the warehouse.

[3] In this situation the test can thus be performed independently of the value of c_i. If, however, the costs of performing the test are different depending on values of c_i, then one may need to introduce a helper condition c_j that models those costs. If required by the diagnostic context, one could make a modification to move the dynamic costs per condition from system-wide to test-specific.

For illustration, let's consider an example connecting to the printing systems described in Section 3, a printing system that has, among others, the following three system conditions:

1. *system_covers_open*, referring to whether the covers of the printing units are on or off. Assuming the system broke down during operation, the initial value is *False* as the covers are closed.[4]
2. *ink_drained*, referring to whether the ink has been drained from the system. Assuming that the system broke down during operation, the ink is still in the system and, therefore, the initial value is *False*.
3. *cooled_down*, referring to whether high temperature subsystems have been given enough time to cool down before service engineers can actually perform measurements in them. As the system was in operation when breaking down, the initial value is *False*.

Given these conditions, and ignoring anything else for the sake of this example, the current system state can be described formally with a function s as follows:

$$\begin{aligned} s(1) &= 0, \\ s(2) &= 0, \\ s(3) &= 0. \end{aligned} \tag{6}$$

To dynamically compute costs, we still need to specify the dynamic cost function c as follows:

$$\begin{aligned} c(1,0) = 50, c(1,1) = 50 \\ c(2,0) = 150, c(2,1) = 50 \\ c(3,0) = 0, c(3,1) = 50 \end{aligned} \tag{7}$$

Note that the values are chosen to roughly represent the complexity of changing the system condition accordingly; they are not based on real data. For example, concerning the second line, we estimate that draining the ink from the system is roughly 3 times simpler than filling it back in. In a real-world scenario, one could choose the numbers to represent actual monetary effort required for the tests.

To finish the example, consider a diagnostic test T_0, which is a measurement of the boiler, a high-temperature subsystem. This test requires removing the system covers as well as waiting for the boiler to cool down:

$$C_0 = \{(1,1), (3,1)\}. \tag{8}$$

Note that condition 2 does not appear in C_0 as we assume, in this example, that it does not matter for T_0 whether the ink has been drained from the system.

Assuming the test has a fixed cost of 100, we compute its total cost given the current system state s:

$$c^s_{total}(T_0) = c_{fix}(T_0) + c(1,1) + c(3,1) = 200. \tag{9}$$

We can see that condition 2 is neglected in this calculation, as it does not matter for T_0. Finally, we update the system state s to s':

$$\begin{aligned} s'(1) &= 1, \\ s'(2) &= 0, \\ s'(3) &= 1. \end{aligned} \tag{10}$$

The crucial observation is that, for example, a test T_1 concerning the boiler as well, with the same preconditions, $C_1 = C_0$, will now be cheaper:

$$\begin{aligned} c^s_{total} = c_{fix}(T_1) + c(1,1) + c(3,1), \\ c^{s'}_{total} = c_{fix}(T_1). \end{aligned} \tag{11}$$

The costs for T_1 are reduced by the amount required to bring the system in the prerequisite state for performing the test.

The example lets us conclude that calculating dynamic costs of diagnostics tests is insightful for making good recommendations to service engineers. In a next step, where we combine entropy (information gain) and dynamic costs, we will see that T_1 becomes more preferable after T_0 has been performed due to the reduced cost.

4.3. Combining Entropy and Cost in Loss

Our approach for combining information gain and cost is to combine them into a single *loss value*, aggregating recommendations that take both criteria into account.

Recalling that, by Eq. (1), we want to select the test T_i that maximizes the information gain, i.e. the difference calculated in Eq. (1). Note that maximizing the difference there is equivalent to minimizing the conditional entropy $E_{M \leftarrow A_i}(\Omega|O_i)$, given that the value $E_M(\Omega)$ is independent of the test T_i. Hence, we can equivalently state Eq. (1) as:

$$T^* = \operatorname*{argmin}_{T_i} E_{M \leftarrow A_i}(\Omega|O_i) \tag{12}$$

This observation is useful in the remainder of this section as we can now formulate our test selection problem as a two-fold minimalization exercise: Given a test $T_i = (A_i, O_i)$, we want to minimize the conditional entropy $E_{M \leftarrow A_i}(\Omega|O_i)$ as well as the cost $c_{total}(T_i)$.

We want to minimize both at the same time, which is not always possible as sometimes the most insightful diagnostic action might also be the most expensive. Hence, we need to balance the two appropriately to achieve a reduction of total diagnostic costs by a balanced reduction of both the expected conditional entropy as well as the cost of the next action.

[4] When one opens the system covers, then one, eventually has to close them again. These cost are covered by our framework in Eq. (7) by specifying the costs for setting this condition from true to false.

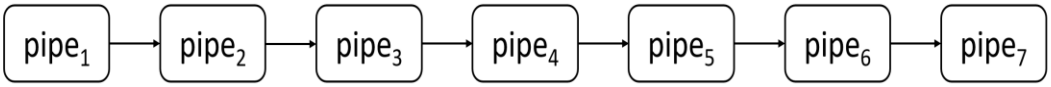


Figure 8. Topology of the system used for the verification. Our software generates the corresponding BN from this System Design Model as outlined in Section 2.2.

Formally, we propose the following formula for aggregating the loss $l(T_i)$ of a diagnostic action T_i:

$$l_b^s(T_i) = b\frac{E_{M\leftarrow A_i}(\Omega|O_i)}{\max_i E_{M\leftarrow A_i}(\Omega|O_i)} + (1-b)\frac{c^s_{\text{total}}(T_i)}{\max_i c^s_{\text{total}}(T_i)}, \quad (13)$$

where the balance $b \in [0,1]$. Note that we omit the superscript s when it is clear from the context. It follows from Eq. (13) that minimizing $l_0(T_i)$ is equivalent to minimizing $c_{\text{total}}(T_i)$ and minimizing $l_1(T_i)$ is equivalent to minimizing $\text{E}(\Omega|T_i)$. Any value strictly between 0 and 1 minimizes a combination of cost and entropy. We normalize both entropy and cost with respect to their maximal values (at the current stage); the range of l_b is thus contained in $[0,1]$. Normalization is necessary to overcome imbalances resulting from the very different scales of entropy and cost (the difference can easily span several orders of magnitude). The latter point is also an advantage we see of our loss function over other approaches, such as minimizing the information gain per cost.

The recommended next diagnostic test is the one that has least loss value:

$$T^* = \operatorname*{argmin}_{T_i} l_b(T_i), \quad (14)$$

where b has been appropriately chosen.

Our suggestion is that b should be chosen in such a way as to minimize the overall expected diagnostic cost of the system. Computing this is a non-trivial task due to the complexity of the systems under consideration. First investigations show that *the right b* is highly unstable and depends not only on the system model but also on the distribution of costs and prior probabilities of the health of components.

We conclude this section by relating the theory back to our example above. We saw above that test T_1 became cheaper after performing test T_0 because they had the same requirements on the system, namely, that the system covers are open (condition 1) and that system has cooled down sufficiently (condition 3). Assume, for now, that the conditional entropy of T_1 was unchanged after performing T_0, and that $b \neq 0$. Given our observation that $c^s_{total}(T_1) < c^{s'}_{total}(T_1)$, we derive

$$l_b^{s'}(T_1) < l_b^s(T_1). \quad (15)$$

Hence, the loss of performing T_1 decreased after performing T_0, and, T_1 is more likely to be recommended as next best diagnostic test. Note, also, that tests that require, e.g., the system covers to be closed, will be more expensive after performing T_0, and are therefore less likely to be recommended as next diagnostic test.

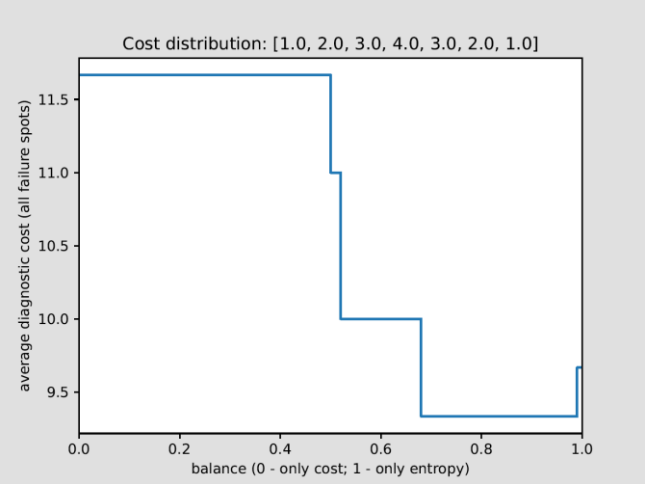
For an initial verification of our approach, we ran a range of simulations to study the impact of including costs on the average overall diagnostic costs. We start with a description of the dummy system on which we ran the simulation, and then describe the simulation and outcomes afterwards.

We consider a dummy system of 7 sequentially connected pipes where water is entering but not leaving the system because one of the pipes leaking (see Figure 8). Under this one-fault assumption, our goal is to identify the leaking pipe with minimal costs. Each of the 7 pipes can be inspected for the presence of water individually, with individual costs assigned to each such inspection, as there might be different degrees of difficulty in reaching the pipes. Failures propagate through the system, i.e., if a pipe does not have water, then all succeeding pipes are also empty.

During the simulation, we fix a distribution of costs for diagnosing each of the 7 pipes. We then consider all possible single faults (i.e. each of the 7 pipes could be the failure pipe) and run a diagnostic procedure for each parameter $b \in [0,1]$ with steps of size 10^{-3}. The procedure stops when the correct fault has been identified with high probability (i.e. > 0.99).[5] With a fixed fault and parameter b, we define the *diagnostic cost* of the procedure to be the sum of the costs of each diagnostic action that was taken in the course of the procedure. Given a fixed parameter b, we calculate the *average diagnostic cost* by averaging out the diagnostic costs for each fault. We present the results of two particularly interesting simulation instances in Figure 9.

In general, we conclude that the optimal choice for parameter b highly depends on the distribution of costs in the systems. The simulations (including those in Figure 9) already indicate that including costs is (in some situations) beneficial over considering only the entropy of a diagnostic test. However, a verification of our methodology in an "industry-as-a-lab" setting is on-going and future work. The latter also includes further mathematical analysis and methods for identifying the optimal parameter b.

[5] By design of our dummy system, this always happens when there is a single fault. Note that we do not consider the case of false positive or false negative test results in this scenario.



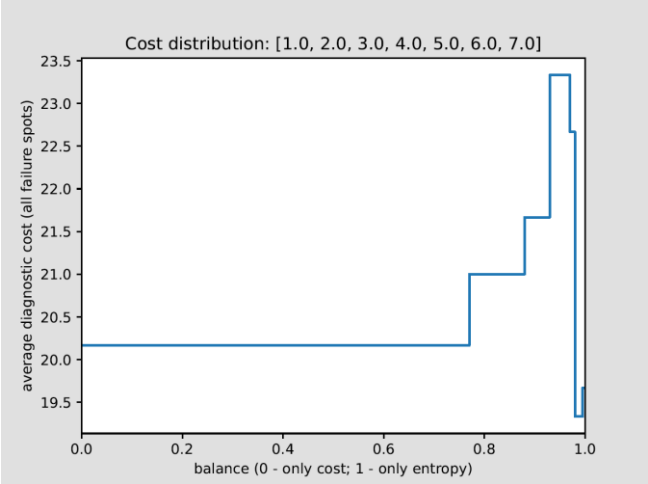


Figure 9. The graphs show the average cost of a diagnosis depending on the parameter b for two example distributions of costs in the system described below. The results indicate (1) that the optimal choice of the parameter b is highly dependent not only on the system but also the cost distribution, and (2) including cost has the potential to reduce average diagnostic cost if the parameter b is chosen correctly.

5. USE CASES

We applied our methodology to two use cases, targeting support service engineers in one, and autonomous decision of surveillance robots in the other. As we illustrate below, the system descriptions that underline our approach focus on different aspects in the respective cases, including behaviors in the latter. This gives us confidence that our methodology is applicable to many scenarios, allowing us to consider active testing in other domains as well.

5.1. Production Printing

Canon Production Printing, part of Canon Inc., designs and manufactures large professional printers. In professional markets, like book on-demand printing, minimal downtime is key to reduce total costs of ownership, with the avoidance of unscheduled downtimes of equal importance. Especially for the first objective, service engineers must quickly diagnose misbehaving printers – a task that is complicated by the high system complexity that resulted from increasing demands on print quality as well as on production throughput. To assist these service engineers, we pioneered and established our methodology on a subsystem of about 50 components and qualitatively assessed the recommendations given by our diagnostic system in different scenarios.

The results exceeded expectations, leading to Canon Production Printing incorporating the method in their service tooling, for which a prototype is shown in Figure 10. Factors in this success were stringent diagnostic conclusions that stem from our diagnostics' basis in system modeling, and the push towards the most likely root cause that follows from the a priori data on component failure likelihoods but also from the tool's ability to propose novel diagnostic strategies that minimize efforts. On the latter, we saw our model ruling out many potential faults with quick active tests that were seemingly unrelated to the current issue, but surprisingly greatly reduced the uncertainty about the system's state.

5.2. Adaptive Behavior in Robot Dogs

The ability to compute which intervention yields best results has applications beyond diagnostics: It also allows to select the response of an autonomous system to circumstances that do not permit successful operations. We investigated this for an autonomous robot dog (TNO, 2022) based on the mobile robot Spot from Boston Dynamics that we develop for several use cases, including indoor search and rescue and inspection of industrial sites.

Equipped with lights, microphones, both visual and acoustic cameras, and gas sensors, this robot navigates on its own, moving in potentially dangerous environments, to find people or objects, e.g., victims of a hazard or leaking pipes.

These search operations are often affected by adverse conditions, like noise from wind, smog, and fog. It is thus not sensible to simply follow prescribed procedures for the inspections or a pre-defined search pattern: the robots must instead actively seek to improve their performance, especially their detection and perception capabilities, to overcome the environmental disturbances. We reached such adaptive behavior with an online diagnosis system that reasons about causes for low performance and computes the most promising action, like moving in the line of the wind to be able to detect gas or out of a zone with high noise to achieve a better performance of an acoustic camera in a search or classification task.

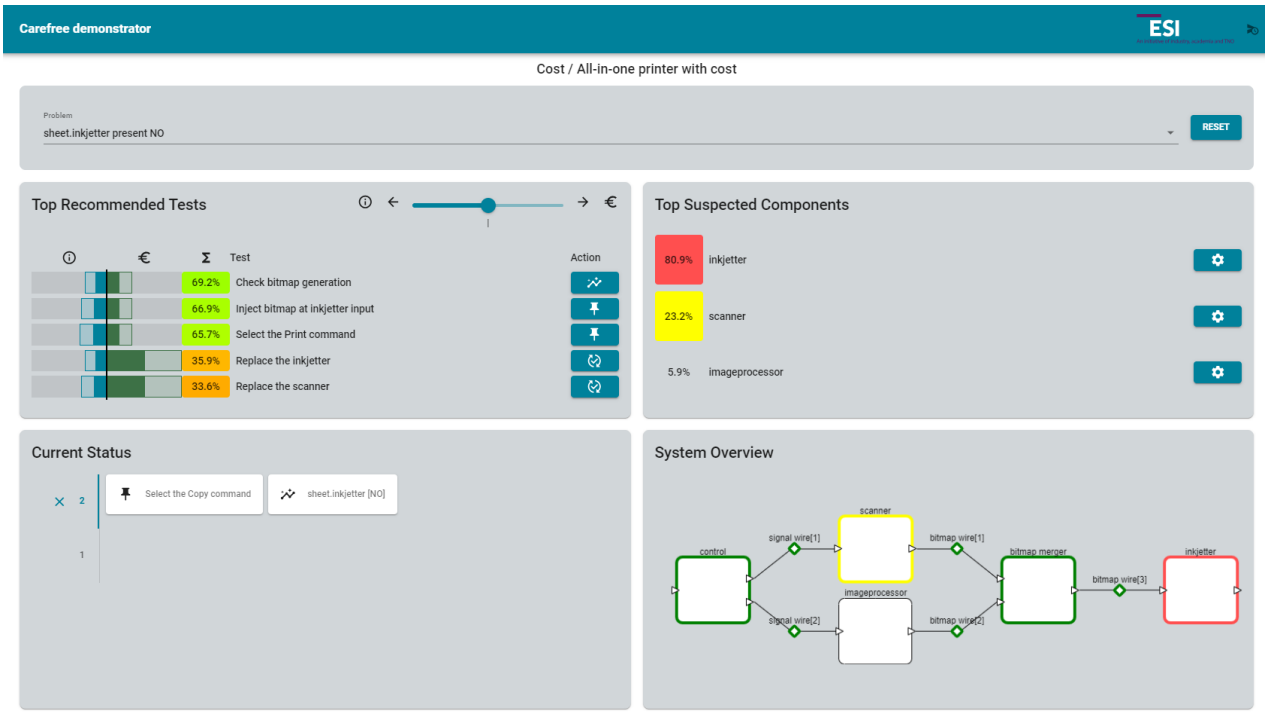


Figure 10. Prototype of the diagnosis application with the reported problem shown on top. On the right, the components are listed in decreasing failure probability with a graphical representation for demonstration purposes. The left middle section lists possible tests in recommended order, allowing the user to vary the cost/entropy trade-off for experimentation via the slider. By selecting a test, the user is presented with the test instructions and the test result entry fields (not shown here). The lower left section shows the history of diagnostic tests that have been conducted thus far and their outcomes. The Bayesian Belief Networks, matching those in Section 3, are generated on the fly during the diagnostic procedure.

The first part of this process, determining the likely cause of low performance, is a classical diagnostic task, even though the system design model that provides for the generation of the differential machine diagnosis differs strongly from the system design schematics that we used in the Canon printing use case. Focusing on the behavior of the robot and the impact of environmental factors on its performance instead, we use a knowledge base along the lines of the concepts shown in Figure 10 as input for the generator.

This knowledge base expresses objects and their possible relationships (Sijs & Fletcher, 2021, 2022), but also possible actions of the robot and expectations on their outcome. Figure 11 provides some examples for this, depicting, e.g., that a gas leak will cause a sound that can be picked up by audio measurements provided that the distance to the sound is not too large, the sound is not covered by noise or other sounds, and the sensor for the audio measurements is in a good state.

Regarding the possible actions of the robot, the knowledge base contains the information that the robot can move itself, thus changing the distance to the gas leak and to interfering sound sources and their impact on the audio measurements, or that it can switch sensors. These actions are interventions, as they change the causality of the overall system, i.e., the robot dog within its environment. This enables the same line of reasoning that we introduced above to determine the most beneficial action to take, only this time by the autonomous robot itself, and not by a human fixing an issue.

Same principle, different process

Changing our task from service diagnostics to active behavior adaptations required a re-interpretation of the reasoning processes and of the elements within them:

The initial diagnostic identification of the components that likely caused a machine to fail is extended towards a broader set of root causes, which now includes environmental impact factors, and moreover reasons for loss of performance instead of strict failures. This extension towards performance diagnostics of a system seen within its environmental context introduces extra factors and enlarges the state-space of the reasoning model but does not pose a fundamental change.

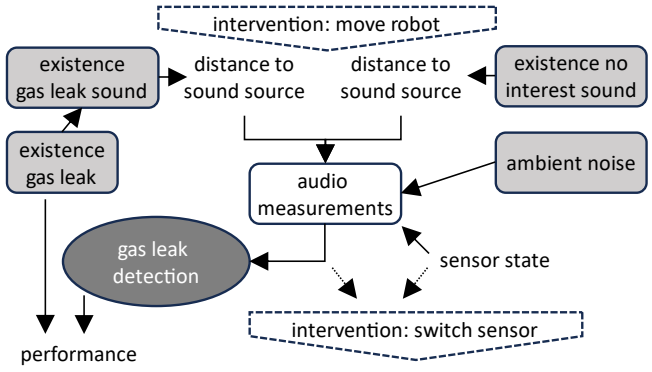


Figure 11. Part of a knowledge base on autonomous robots

Once a likely root cause of low performance is identified by the robot's reasoning, the selection of the appropriate action is based on calculations of the likely effect of that particular intervention, essentially asking <if the robot takes this action, would the problem go away?> This changes, at first glance, the calculation of the expected reduction of entropy following diagnostics towards the calculation of the expected reduction of performance loss. However, it maps to an active diagnostic process that only allows diagnostic tests of types II, III, and IV (i.e., interventions) and keeps going through interventions until it finds a configuration in which the robot performs.

An important aspect in this is that our methodology includes the information gained in previous attempts in its diagnostic reasoning. In the case of the autonomous robot, this solves a potential issue of machine reasoning: It prevents – figuratively and possibly literally – that the robot runs in circles, i.e., that it alternates between actions that look promising for solving the performance issue but actually fail to do so. Such a situation, in which multiple computational attractors must be interpreted as in-between steps towards a solution that should only be tried once, would not confound a human diagnosis expert – but a simpler reasoner trying to handle an open world might fail by switching between them.

6. CONCLUSION AND FUTURE WORK

Being able to efficiently conduct active diagnostics with intervening tests is crucial for asset life-cycle management. Based on the theory of counter factual reasoning, we can now integrate such interventions in industrial practice and tool support, where our results indicate that the method enables a structured way of efficiently and effectively deciding among possible diagnostic actions. Nevertheless, the decision support as outlined in Section 4 can be refined to further scale up to complex scenarios.

Firstly, Eq. (1), which is inspired by existing service manual procedures, assumes a fixed relation between an action A_i and observation O_i. In theory, however, we could do any action A_i and then observe some O_j, leading to

$$T^* = \underset{T_i}{\operatorname{argmin}}\, l_b(A_i, O_j), \tag{16}$$

i.e., recommending the combination of observation and action that has least loss. However, many of the pairs (A_i, O_j) make no sense in a practical situation, so the computational complexity might not be worth the additional information gain – but deciding this depends on many factors, including business reasoning, and warrants future investigations.

Secondly, it is not clear how to choose the optimal balance value b in the loss function l_b to reduce overall diagnostic cost. Initial investigations show that the optimal b varies greatly with the selected model, distribution of costs, and prior failure probabilities. Consequently, we will investigate further optimization of this value and, potentially, related loss functions in future work as well.

ACKNOWLEDGEMENT

The research is carried out as part of the Carefree project under the responsibility of TNO-ESI with Canon Production Printing as the carrying industrial partner. The Carefree research is supported by the Netherlands Organisation for Applied Scientific Research TNO as part of the Appl.AI program, which is also supporting the research's application on the autonomous robot as part of the SEAMLESS and SNOW projects.

REFERENCES

Balke A, Pearl J, (1994). Probabilistic evaluation of counter-factual queries. *Annual AAAI Conference on Artificial Intelligence AAAI-94* (230-237), Seattle, USA.

Bayes Server, www.bayesserver.com, retrieved 24-03-2023.

Barbini, L., & Borth, M. (2019). Probabilistic Health and Mission Readiness Assessment at System-Level. *11th Annual Conference of the PHM Society*, 11(1) Scottsdale, USA.

Barbini, L., Bratosin, C., & van Gerwen, E. (2020). Model-based diagnosis in complex industrial systems. *5th European PHM Society Conference*, 5(1), 8, Turin, Italy.

Borth, M., & von Hasseln, H. (2002). Systematic generation of Bayesian networks from systems specifications. *IFIP 17th World Computer Congress on Intelligent Information Processing* (155-166), Montréal, Canada.

van Gerwen, E., Barbini, L. & Borth, M. (2023). Differential Diagnosis with Active Testing. *Asia-Pacific Conference of the PHM Society*, vol. 4, no. 1, Tokyo, Japan.

de Kleer, J., Raiman, O. & Shirley, M. (1992). One Step Lookahead is Pretty Good, In Hamscher, W., de Kleer, J. and Console, L. (Eds), *Readings in Model-Based Diagnosis*, (138-142). Morgan Kaufmann.

Heckerman, D., Mamdani, A. & Wellman, M.P. (1995). Real-world applications of Bayesian networks. *Communications of the ACM*, 38.3, 24-26.

Jensen, F.V. & Nielsen, T.D. (2007). Bayesian Networks and Decision Graphs, Springer, New York, USA.

Oladyshkin, S. & Nowak, W. (2019). The Connection between Bayesian Inference and Information Theory for Model Selection, Information Gain and Experimental Design. *Entropy*, vol. 21, no. 11, doi: 10.3390/e21111081.

Pearl, J. (1986). Fusion, propagation, and structuring in belief networks. *Artificial intelligence*, 29(3), 241-288.

Pearl, J. (2009). Causality. Cambridge University Press.

Pearl, J. & Mackenzie, D. (2018). The Book of Why: The New Science of Cause and Effect. Basic Books.

Pfeffer, A., Koller, D., Milch, B. & Takusagawa, K.T. (1999). SPOOK: A System for Probabilistic Object-Oriented Knowledge Representation. *Fifteenth Conference on Uncertainty in Artificial Intelligence* (541-550), Stockholm, Schweden.

Ricks, B. W., & Mengshoel, O. J. (2009). Methods for probabilistic fault diagnosis: An electrical power system case study. Annual Conference of the PHM Society 2009, 1(1), San Diego, USA.

Sijs, J. & Fletcher, J. (2021). A knowledge base for robots to model the real-world as a hypergraph. *Fifth IEEE International Conference on Robotic Computing (IRC)*, (119-120),Taiwan. doi: 10.1109/IRC52146.2021.00027.

Sijs, J. & Fletcher, J. (2022). On a hypergraph structuring semantic information for robots navigating and conducting their task in real-world, indoor environments. *26th International Conference on Methods and Models in Automation and Robotics* (430-435), Poland. doi: 10.1109/MMAR55195.2022.9874265.

TNO (2022). Situational awareness in robot dogs. www.tno.nl/en/digital/artifical-intelligence/safe-autonomous-systems/situational-awareness-robot-dogs/