Pervasive Diagnosis

Lukas Kuhn, Bob Price, Minh Do, Juan Liu, *Member, IEEE*, Rong Zhou,
Tim Schmidt, and Johan de Kleer, *Member, IEEE*

***Abstract*—In model-based production, a planner uses a system description to create plans that achieve production goals. The same description can be used by model-based diagnosis to infer the condition of components from sensor data. When production is realized by a sequence of plans, prior work has demonstrated that diagnosis can be used to adapt the plans to compensate for component degradation. However, the sources of diagnostic information are severely limited. Diagnosis must either make inferences from observations during production over which it has no control (passive diagnosis), or production must be halted to introduce diagnostic-specific plans (explicit diagnosis). We observe that the declarative nature of the model-based approach allows the planner to achieve production goals in multiple ways. This flexibility is exploited by a novel paradigm, i.e., *pervasive (active) diagnosis*, which constructs *informative production plans* that simultaneously achieve production goals while uncovering additional diagnostic information about the condition of components. We present an efficient heuristic search for these informative production plans and show through experiments on a model of an industrial digital printing press that the theoretical increase in long-run productivity can be realized on practical real-time systems. We obtain higher long-run productivity than a decoupled combination of planning and diagnosis.**

***Index Terms*—Active probing, control, diagnosis, model-based reasoning, planning, scheduling.**

I. Introduction

AUTOMATED diagnosis is a key strategy for improving the reliability and maintainability of complex systems; it reduces labor overhead and leads to more rapid repairs. In remote applications, such as autonomous spacecraft operation, automated diagnosis can be used when human experts are unavailable. While automated diagnosis of complex systems has many advantages, it is not free. In high-speed mass production settings, stopping production for automated diagnosis, even for a brief interval, incurs significant costs. In demanding industrial applications, every unit of lost production is significant. Reducing these losses by small amounts can make the difference between having a competitive product and bankruptcy. We extend the benefits of automated diagnosis to demanding real-time industrial production through a new active diagnosis paradigm, i.e., *pervasive diagnosis*, which optimally achieves as much diagnosis as possible during production. In general, a system implements active diagnosis when it is able to choose actions to be performed on itself in order to improve its understanding of the condition of its components. Pervasive diagnosis is a new active diagnosis paradigm in which production is actively manipulated to maximize diagnostic information. Active diagnosis and production can, therefore, occur *simultaneously* leading to higher long-run productivity than passive diagnosis or alternating explicit diagnosis with production. The integration of diagnostic goals in the production strategy results in *informative production*. The advantages of pervasive diagnosis are demonstrated on a high-speed multiengine printing platform.

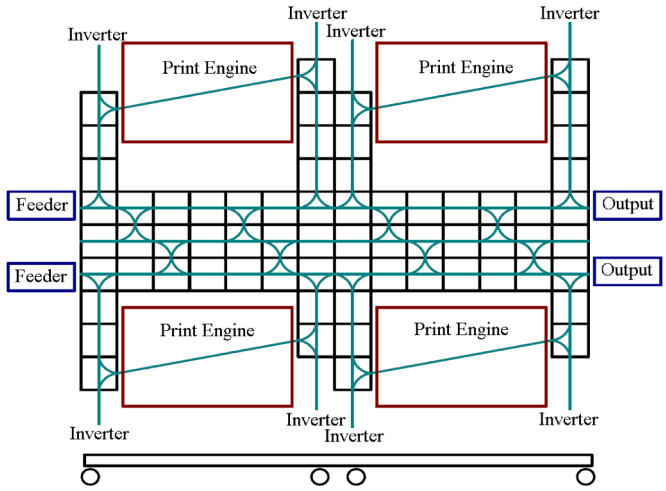

Fig. 1. Four-engine prototype printer built at the PARC with over 170 individually controlled modules.

We develop a formal framework for the production versus diagnosis tradeoff in the pervasive diagnosis paradigm. We start with a description of our primary testbed, i.e., the high-speed digital printing press system shown in Fig. 1, and how it is controlled today by a model-based planner. We then show how this framework is used to construct an efficient automated diagnosis system for our industrial digital printing press testbed. We then examine our framework from a theoretical perspective in order to identify the conditions under which pervasive diagnosis is applicable and characterize the likely gains from this approach. The diagnosis system is demonstrated on a number of scenarios to fully characterize the performance of pervasive diagnosis on realistic problems. We conclude this paper with discussion of related and future work.

Manuscript received May 3, 2008. Date of publication July 26, 2010; date of current version August 18, 2010. This paper was recommended by Guest Editor P. Struss.

L. Kuhn and T. Schmidt are with the Palo Alto Research Center, Palo Alto, CA 94304 USA, and also with the Technical University of Munich, 80333 Munich, Germany (e-mail: lukas.kuhn@parc.com; tim.schmidt@parc.com).

B. Price, M. Do, J. Liu, R. Zhou, and J. de Kleer are with the Palo Alto Research Center, Palo Alto, CA 94304 USA (e-mail: robert.price@parc.com; minh.do@parc.com; Juan.Liu@parc.com; Rong.Zhou@parc.com; johan@dekleer.org).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TSMCA.2010.2052040

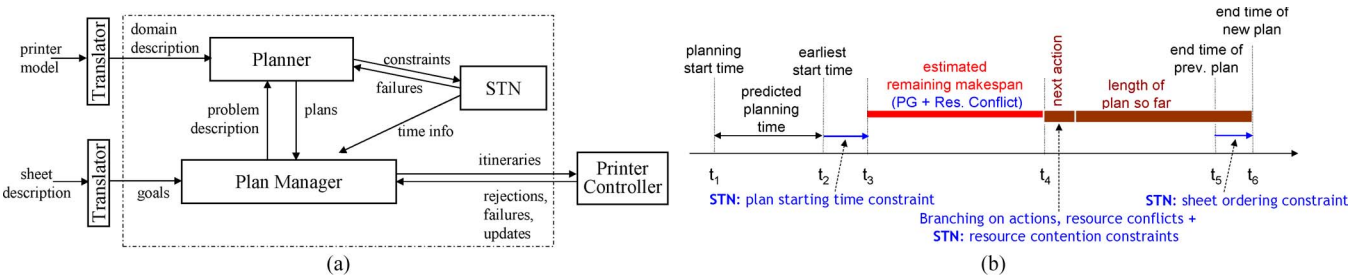

Fig. 2. PARC's tightly integrated planning and scheduling approach. (a) Overall architecture indicated by the dashed box. (b) Important time points for evaluating a plan.

II. Model-Based Planner-Controlled Prototype Digital Printing Press

A modular printer can be seen as a network of paper paths linking multiple printing engines. Fig. 1 shows a schematic side view of the four-engine prototype printer built at the Palo Alto Research Center (PARC). It has over 170 independently controlled modules and many paper paths linking the paper feeders to printers and the possible output trays. Multiple feeders allow blank sheets to enter the printer at a high rate, and multiple output trays allow several jobs to run simultaneously. Having redundant paths through the machine enables graceful degradation of performance when modules fail. Each module has a limited number of discrete actions it can perform, and for many of these actions, the planner is allowed to vary their duration.

Controlling these systems requires planning and scheduling a series of jobs that arrive asynchronously over time. These printers may run at high speed (up to several hundreds of pages per minute) continuously. Each job request completely describes the attributes of its sheets. The plan for printing each individual sheet is a linear sequence of actions. There may be many sequences of actions that can be used to print a given sheet. For example, in Fig. 1, a blank sheet may be fed from any of the two feeders, then routed to any one of the four print engines (or through any two of the four engines in the case of duplex printing), and then to the correct output tray. This online planning problem is complicated by the fact that many sheets are in-flight simultaneously, and the plan for the new sheet must not interfere with those sheets. Moreover, plan synthesis and plan execution are interleaved in real time. Since we minimize wall-clock end time, the speed of the planner itself affects the value of a plan.

A. Planning Architecture

A modular printer resembles any product manufacturing plant, with raw materials (blank sheets) entering at the plant's inputs, being routed through different machines that can change the properties of the materials, and the final products (printed sheets) being collected at the outputs. This domain also has certain properties of package routing or logistics problems with cross-goal resource constraints, which represent a wide range of online decision-making settings, such as multirobot coordination.

Fig. 2(a) shows the core architecture of the planner and how it communicates with the printer controller. The major components are outlined in the next sections. The overall objective is to minimize the end time of all known sheets, ranging over all current print jobs. We approximate this by optimally planning for the current sheet given the resource allocations over time for all previous sheets.

B. Temporal Constraints and Plan Management

Temporal constraints are maintained using a simple temporal network (STN) [1], which is represented by the box named STN in Fig. 2(a). The network contains a set of time points t_i and constraints between them of the form $l_b \leq t_i - t_j \leq u_b$. The time points managed by the STN include action start and end times, as well as resource allocation start and end times, with the following constraints: 1) wall-clock action start time; 2) the range of action start and end times; 3) constraints between action start time and resource allocation by that action; and 4) conflicts for various types of resources. For constraint propagation, we use a variation of the arc consistency algorithm [2].

As the planner uses an A^* search strategy that maintains multiple open search nodes, a separate STN is associated with each node. Temporal constraints are added to the appropriate STN when a search node is generated. Whenever a new constraint is added, constraint propagation tightens the upper and lower bounds on the domain of each affected time point.

The large number of potential plans for a given sheet and the close interaction through resource conflicts among plans for different sheets mean that it is better to process scheduling constraints during the planning process. The planner uses state-space regression to plan each sheet, but maintains as much temporal flexibility as possible in the STN using the partial orders between different actions in plans for different sheets. Therefore, it can be seen as a hybrid between state-space search and partial-order planning.

The plan manager monitors execution, releases plans to the printer controller, and propagates resource commitments back to the planner.

C. Planning Individual Sheets

When planning individual sheets, the regressed state representation contains the (possibly partially specified) state of the sheet. Best-first search (BFS) is used to find the optimal plan for the current sheet, in the context of all previous sheets. Because it must not interfere with existing plans, the state contains information both about the current sheet and previous plans.

More specifically, the state is a 3-tuple $\langle Literals, STN, R\rangle$, where $Literals$ describes the regressed logical state of the current sheet, STN contains all known time points for the state with the current constraints among them, and the resource profile R is the set of current resource allocations, representing the commitments made to plans of previous sheets and the partial plan of the current sheet. After the optimal plan for a sheet is found, the resource allocations and STN used for the current plan become the basis for planning the next sheet.

D. Optimizing Productivity and Heuristic Estimation

Our overall objective is to minimize the earliest possible end time for all planned sheets. In essence, this objective function minimizes the total time to finish all planned sheets and, thus, maximizes the printer's productivity. To support this, the primary criterion for evaluating the promise of a partial plan is the estimate of the earliest possible end time of the partial plan's best completion. To estimate this quantity, we compute a simple lower bound on the additional makespan required to complete the current regressed plan. This heuristic value is indicated in Fig. 2(b) by *estimated remaining makespan*. It is inserted before the first action in the current plan (t_4) and after the plan's earliest start time (t_2). By adding the constraint $t_2 < t_3$, the insertion may, thus, change the end time of the plan. It may also introduce an inconsistency in the temporal database, in which case we can safely abandon the plan. Given that the current plan should end after the end time of all previous sheets in the same print job $(t_5 < t_6)$, our objective function is to minimize t_6 without causing any inconsistency in the temporal database. We break ties in favor of smaller predicted makespan $(t_6 - t_3)$ and then larger currently realized makespan $(t_6 - t_4)$. This is analogous to breaking ties on $f(n) = g(n) + h(n)$ in BFS with larger $g(n)$. Because our heuristic is admissible, the plan found is optimal according to our objective function.

To estimate the duration required to achieve a given set of goals G from the initial state, we do dynamic programming on the planning graph [abbreviated as PG in Fig. 2(b)], in a manner similar to Temporal GraphPlan [3]. To compute more accurate heuristic estimates in the presence of significant resource contention, we take resource conflicts into account [4].

III. Diagnosis and Production

Given the planner-controlled system as described in Section II, we are interested in adding a diagnosis capability without disrupting production. Our framework is based on the conception of active diagnosis. To perform active diagnosis, the model-based planner controlling the production system may also be designed to generate diagnosis plans. The plan influences the system's evolution through state space and possibly determines the measurements that will be made. Different from test generation [5], our approach simultaneously achieves production goals. Executing the plan on the system results in a vector of observations. A sequence of plans will result in a sequence of observation vectors.

The digital printing press may run for hundreds of thousands of sheets before encountering any errors, and, furthermore, most of the errors are intermittent. This motivates some of the design choices underlying our approach. First, given the very large numbers of events, it is unrealistic to retain full sensor information from previous sheets. Therefore, we use counters to track failure events. Second, we are most concerned with intermittent errors as they cause most of the production halts and troubleshooting time. Third, unlike digital circuits, damaged sheets cannot be masked. Thus, any individual component failure will cause a production failure. We call this the *catastrophic* failure assumption. Finally, our prototype printer incorporates multiway internal redundancy, which we exploit heavily.

We assume the system to be detectable [6] (or diagnosable [7]) and isolatable [8]. The central idea is that there always exists a sequence of plans and a resulting sequence of observations that provides information whether the system is faulted and in case of a failure to disambiguate between all possible fault causes. A system that is isolatable is one in which there exists a sequence of plans such that the sequence of observation vectors for a machine with some fault f_i will differ from the observations for a system with any other fault f_j, $i \neq j$. For isolatable systems, we note that the ability to isolate a fault is still a function of the plans chosen. For example, it can reroute sheets to probe suspected modules.

IV. Information Gain During Production

Production is said to be *informative* if it contributes to clarifying ambiguities in the diagnosis. Not all plans are equally informative regarding diagnosis. We use mutual information to measure the information content. It is a metric from information theory, which is commonly used for characterizing the performance of classification and data compression [9].

We use the standard notation, with uppercase symbols denoting random variables and lowercase symbols denoting a particular realization. Let X be the underlying diagnostic state and Y be the observation. For example, the bit vector $X = 011000$ indicates that only the second and third modules are faulted, and the observation of a damaged output is $Y = 1$ and is $Y = 0$ for a good output. Note that X and Y are both random variables. The mutual information between X and Y is defined as [9]

$$I(X;Y) \triangleq \Sigma p(x,y)\left[\log_2 \frac{p(x,y)}{p(x)p(y)}\right]. \tag{1}$$

Conceptually, it measures the bits of information that the observation Y tells about the underlying diagnosis state X. It is nonnegative and is equal to zero if and only if X and Y are independent, in which case, measuring any value of Y has no implication on refining the underlying diagnosis X and, hence, has zero information content. In practice, we should avoid such an irrelevant observation, but rather make an observation that reveals as much information as possible regarding X.

In diagnostic tasks, observations are made from plan executions. Hence, Y is implicitly parameterized by the plan. To emphasize this, we use Y_P, with P denoting the production plan. The goal for plan selection is to find P such that $I(X;Y_P)$ is maximized. We discuss the search for plans in a later section. It is straightforward to compare plans. For example, given two

choices P_1 and P_2, we say that P_1 is more informative and preferable to P_2 if

$$I(X;Y_{P_1}) > I(X;Y_{P_2}). \tag{2}$$

Mutual information $I(X;Y_P)$ measures the reduction of uncertainty in X when observing Y_P, i.e.,

$$I(X;Y_P) = H(X) - H(X|Y_P) \tag{3}$$

where $H(X)$ is the entropy of X, and $H(X|Y_P)$ is the entropy of X conditioned on observing Y, i.e., the "remaining uncertainty" after the observation. Maximizing $I(X;Y_P)$ is equivalent to minimizing $H(X|Y_P)$. This is equivalent to saying that we would like to select the plan, which leaves as little uncertainty as possible. Entropy (or conditional entropy) is a well-accepted and widely used metric for uncertainty. It also serves as a bound: to diagnose a problem with entropy of h bits and come to a deterministic conclusion, the number of tests needed, on average, is at least h.

To calculate mutual information, we take advantage of the symmetry of mutual information, i.e., $I(X;Y_P) = I(Y_P;X)$. The amount of information that Y_P tells about X is equal to the amount that X tells about Y_P. Exchanging X and Y_P in (3), we have

$$I(X;Y_P) = H(Y_P) - H(Y_P|X). \tag{4}$$

Although (3) and (4) are equivalent, the latter is often much easier to compute. In a diagnostic problem, we have a current diagnosis $p(\mathbf{x})$ and an observation likelihood $p(y_P|\mathbf{x})$; hence, the second term $H(Y_P|X)$ is easy to compute. The other way, i.e., $H(X|Y_P)$ in (3), involves the posterior belief $p(\mathbf{x}|y_P)$, which is much harder to compute. In this paper, we will be using (4) for evaluation.

Mutual information has been used as an evaluation and selection criterion in a number of applications. For example, [10] uses mutual information to decide which sensors to activate in the context of tracking a moving target. Similarly, [11] uses mutual information to control a fleet of robots, sending robots to most advantageous locations. In this paper, we extend this framework to enable the integration of planning and diagnosis.

The question is how to diagnose a system when a fault has been observed? A common divide-and-conquer scheme is to devise a plan P that includes only half of the modules. If the plan observes a fault, that means P contains the fault, and the other half that P excludes is cleared of suspicion. If the plan is successful, then P is cleared, and the fault must be in the other half. This way, every plan dissects the diagnosis space by half.

Here, we focus on the task of diagnosing a single intermittent fault to illustrate the idea of choosing the most informative production plan. In the single-fault example, the diagnosis space is linear in the number of modules, i.e., $X = \{1, 2, \ldots, M\}$. We further assume that if module i has a fault and contributes to some plan, it will exhibit aberrant behavior with some probability q_i, i.e., the *intermittency rate*, resulting in plan failure.

The divide-and-conquer scheme can be generalized via the mutual information criterion. In the product plant abstraction, each faulty module can damage the product with an intermittent probability q if it is included in the production plan P, i.e., with the observation likelihood. For single faults

$$p(y_P|X=i) = \begin{cases} 0, & \text{if } y=1 \text{ and } i \notin P \\ 1, & \text{if } y=0 \text{ and } i \notin P \\ q_i, & \text{if } y=1 \text{ and } i \in P \\ 1-q_i, & \text{if } y=0 \text{ and } i \in P. \end{cases} \tag{5}$$

Define $H_q^{(i)}$ as the entropy corresponding to the binomial distribution q_i, i.e., $H_q^{(i)} \triangleq -[q_i \log q_i + (1-q_i)\log(1-q_i)]$. The mutual information can be evaluated as

$$I(X;Y_P) = [-y_0 \log y_0 - y_1 \log y_1] - \sum_{i \in P} p_i H_q^{(i)} \tag{6}$$

where y_0 is the probability of observing a success, and y_1 is the probability of observing a failure. The derivation follows from (4). The first term (the bracketed term) is $H(Y_P)$, and one can easily verify that the second term is $H(Y_P|X) = \sum_{i \in P} p_i H_q^{(i)}$.

An interesting special case is when all faults are persistent, i.e., $q_i = 1$ for all i. In this case, all $H_q^{(i)} = 0$, and the second term in (6) vanishes. The mutual information is, hence, only $-y_0 \log y_0 - y_1 \log y_1$, maximized when $y_0 = y_1 = 0.5$. Thus, the best plan is the one that touches closest to half the probability mass, i.e.,

$$P_{DX}^* = \underset{P \in \mathbb{P}_{I \to G}}{\arg\min} \, | \, p_t \left(AB(P) \right) - T | \tag{7}$$

where the target $T = 0.5$, $\mathbb{P}_{I \to G}$ being the set of plans that achieves the production goal G. This is a generalization of the divide-and-conquer strategy above. An algorithm for finding plans closest to the target T is presented later.

In the intermittent fault case, the second term is nonzero and can be considered as a "correction" term due to the intermittency. When all the modules have the same q_i value (this is likely in the paper path modules that all have the identical design), the mutual information can be further simplified. It can be evaluated as the function of a single variable $w = \sum_{i \in P} p_i$, i.e.,

$$-[wq \log wq + (1-wq)\log(1-wq)] - wH_q. \tag{8}$$

Given any plan P, we can evaluate the accumulative probability w and the corresponding mutual information value. Similarly, we can optimize the search for the most informative path P or, equivalently, for the optimal w value. Through simple calculus, we obtain the optimal w as

$$w = \frac{1}{q(2^{H_q/q}+1)} = \frac{1}{\left(\frac{1}{1-q}\right)^{\frac{1-q}{q}} + q}. \tag{9}$$

When $q \to 0$, w is asymptotically approaching $1/e$. This can be easily verified from the limit

$$\lim_{q \to 0} \left(\frac{1}{1-q} \right)^{\frac{1}{q}} = e. \tag{10}$$

For $q \in (0, 1]$, w takes values from $1/e$ to 1/2. Thus, in the case of intermittent faults, we see that the target T is determined by the q_i's of the modules and lies between $1/e$ to 1/2, unlike

$T = 1/2$ for persistent faults. The concept can be generalized to the diagnosis of multiple faults. Multiple-fault diagnosis can be done dynamically [12], [13] from imperfect tests [13].

V. Representing Information

Given a planner-controlled system, we can enable the planner to optimize for both productivity and maximal information gain. Before we present this integration, we introduce our representation. We model such a system as a state machine with states $\mathcal{S}$, actions $\mathcal{A}$, and observables $\mathcal{Y}$. The system is controlled by a plan P consisting of a sequence of actions $a_1, a_2, \ldots, a_n$ drawn from the set of actions $\mathcal{A}$. Executing the actions of plan P results in an observation Y. We assume a system with sensors only at the output.

When a plan fails to achieve its goal [$AB(P)$, thus $Y = 1$], the system attempts to gain information about the condition of its components. Information about the system is represented by the diagnosis engine's belief in the hypotheses. A hypothesis is a conjecture about what caused the problem. It is represented as a set of actions believed to be abnormal, i.e.,

$$X = \{a | a \in \mathcal{A}, AB(a)\}. \tag{11}$$

The set of all possible hypotheses is denoted by $\mathcal{X}$, the power set of $\mathcal{A}$. We distinguish one special hypothesis: the *no fault* hypothesis X_0 under which all actions are not abnormal.

The beliefs at time t are represented by a probability distribution over the hypothesis space $\mathcal{X}$, $p_t(X)$. The beliefs are updated by a diagnosis engine using past observations and Bayes' rule to obtain a posterior distribution over the unknown hypothesis X given observation Y and plan P, i.e.,

$$p_t(X|Y, P) = \alpha p_t(Y|X, P) p_t(X).$$

As the machine may run almost constantly, we maintain counters to summarize results of all prior plan executions. We adopt the counting convention from [14]:

- m_{11}: the number of plans where m was used and failed;
- m_{10}: the number of plans where m was used and succeeded;
- m_{01}: the number of plans where m was not used and failed;
- m_{00}: the number of plans where m was not used and succeeded.

Consider the case of a single intermittent fault. Let q_i be the probability that module i has a faulty output. Given the likelihood equation (5), after t iterations [except in the case when $m_{01} > 0$ in which case $p_t(X|Y, P) = 0$]

$$p_t(X|Y, P) = (1 - q_i)^{m_{10}} q_i^{m_{11}}.$$

This approach directly expands to any combination of persistent and intermittent faults [15].

A. Failure Probability of a Plan

The first task is to be able to predict the failure probability associated with a given plan $P = a_1, a_2, \ldots, a_n$. We denote the set of unique actions in a plan as

$$\mathcal{A}_P = \bigcup_{a_i \in P} \{a_i\}. \tag{12}$$

Formally, under the catastrophic fault assumption, the plan will be abnormal $AB(P)$ if any action in the plan is abnormal, i.e.,

$$AB(P) \Leftrightarrow AB(a_1) \vee \cdots \vee AB(a_n). \tag{13}$$

The predicted probability of a plan being abnormal is a function of the probabilities assigned to all relevant hypotheses. The set of hypotheses that bear on the uncertainty of the outcome of plan P is denoted $\mathcal{X}_P$ and is defined as

$$\mathcal{X}_P = \{X | a \in X,\ a \in \mathcal{A}_P,\ X \in \mathcal{X}\}. \tag{14}$$

Therefore, plan P will fail whenever any member of $\mathcal{X}_P$ is true, i.e.,

$$AB(P) \Leftrightarrow x_1 \vee x_2 \vee \cdots \vee x_m, \qquad \text{where } x_j \in \mathcal{X}_P. \tag{15}$$

The probability of a plan failure $p_t(AB(P))$ is defined as the sum of all probabilities of hypotheses in which the plan fails, i.e.,

$$p_t\left(AB(P)\right) \triangleq \sum_{X \in \mathcal{X}_P} p_t(X). \tag{16}$$

B. Learning

The majority of the faults in the printer are intermittent. Unfortunately, it is very difficult to obtain prior values for the rate of intermittency (the q_i's). In addition, the q_i values can vary. Therefore, we do not rely on prior values for q_i and instead use a Bayesian filter to estimate the values for q_i. Fortunately, a reasonable estimate can be obtained from the counters m_{ij} (counters are initialized to 1), i.e.,

$$q_i = \frac{m_{11}}{m_{11} + m_{10}}. \tag{17}$$

Thus, in the intermittent single-fault case

$$p_t(X|Y, P) = \alpha \left[\frac{m_{10}}{m_{11} + m_{10}}\right]^{m_{10}} \left[\frac{m_{11}}{m_{11} + m_{10}}\right]^{m_{11}} p_0(X) \tag{18}$$

where α is the normalization factor.

VI. Multiobjective Search for Informative Production Plans

As outlined in the earlier sections of this paper, our objective is to enable the planner to optimize for both productivity and maximal diagnostic information. For this objective, the optimal plan is the one with failure probability as close as possible to the target value T defined in Section IV. Here, we will discuss the two main issues related to finding such a plan:

- how to change the node ranking functions in the current BFS framework to account for our new diagnosis-related objective functions;

- how to calculate the heuristic guiding the BFS framework with the new objective functions.

In BFS, planning search nodes are ranked according to measurements that are related to the objective functions. Recall from the previous section on planning for productivity, the measure of printer productivity is the estimated time t to finish executing all plans. The lower ranking functions (or tie breakers) such as estimation of total plan execution time or the length of the partial plan found so far are then used to improve the search process. Similarly, for the new objective function of maximizing diagnostic information, we also have the set of ordered measurements to rank which nodes should be picked next in the open search queue. For this objective function, the most important factor is the information gain determined by the distance between the predicted plan failure probability and the target probability [see (7)]. To improve search efficiency, we also use several lower ranking functions to break ties. The details and the intuition behind those ranking functions will be described toward the end of this section after we discuss how we estimate v.

Given that the two objective functions, i.e., 1) maximizing productivity (minimizing plan finishing time, denoted $f_{\text{ProductionTime}}$) and 2) maximizing diagnosis information (minimizing ambiguity, denoted $f_{\text{DXambiguity}}$), are independent, improving the final plan quality along one dimension can worsen the quality according to the other. Therefore, at the top level, we allow the user to specify the "weights" for each objective function. Using a weighted sum to combine different nondependent objective functions is a standard approach, and in our case, the top-level objective function f is

$$f = \gamma f_{\text{DXambiguity}} + (1-\gamma) f_{\text{ProductionTime}} \tag{19}$$

where the γ value is user-defined and is sent by the diagnoser to the planner whenever the system is switched to the pervasive diagnosis mode. In most of our experiments, we used the value $\gamma = 1$. Thus, we favor maximally informative plans and break ties on productivity. The complete order set of ranking functions are listed at the end of this section.

A. Diagnostic Plan Heuristics

Here, we discuss our implemented approach for estimating the value $v = |p_t(AB(P)) - T|$ and, along the way, define the lower ranking functions (i.e., tie breakers) that allow more efficient search. To estimate the value of v, we use dynamic programming to incrementally update the estimate of the upper and lower bound values on $p_t(AB(P))$ using the action failure probabilities $p_t(a)$ provided by the diagnoser. We illustrate this incremental process with an example.

Example: Consider the graph in Fig. 3, which represents legal action sequences or possible plans that can be executed on a manufacturing system. A system plan starts in the initial state I and follows the arcs through the graph to reach the goal state G.

Suppose that we execute a plan that moves the system through the state sequence $[I, A, C, G]$. The sequence is shown as a shaded background on the relevant links in the figure. Assume that the plan resulted in an abnormal outcome. Unknown

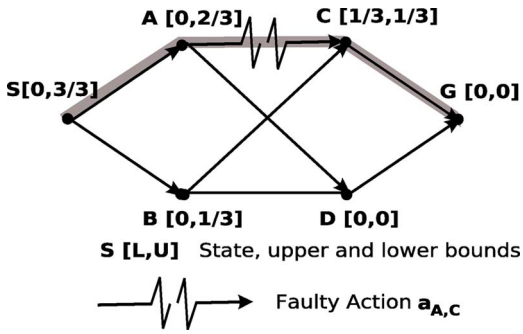

Fig. 3. Machine topology places limits on what can be learned in the suffix of a plan.

TABLE I
PROBABILITY OF HYPOTHESES (SINGLE FAULT)

Hypothesis	$\{a_{I,A}\}$	$\{a_{A,C}\}$	$\{a_{C,G}\}$
Probability	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$

to the diagnosis engine, the abnormal outcome was caused by action $a_{A,C}$. Assume that the fault is persistent. A diagnosis engine would now suspect all of the actions along the plan path to be faulty. We would have three hypotheses corresponding to suspected actions: $\{\{a_{I,A}\}, \{a_{A,C}\}, \{a_{C,G}\}\}$. In the absence of additional information, we might assign them equal probability (see Table I).

The graph structure and probability estimates can be used to construct heuristic bounds on the uncertainty that can be contributed to a plan by any plan suffix. We build up the heuristic backward from the goal state G (the right side of the figure). Consider action $a_{D,G}$ leading from state D to the goal state G in Fig. 3. Action $a_{D,G}$ was not part of the plan that was observed to fail, so it is not a candidate hypothesis. Under the single-fault hypothesis, it has probability zero of being faulted. If we extend any prefix plan ending in state D with action $a_{D,G}$, we do not increase the failure probability of the extended plan because the action $a_{D,G}$ has probability zero of being abnormal. There are no other possible plans from D to G, so both the upper and lower bounds for any plan ending in state D are zero.

Similarly, we determine that state B also has a lower bound of zero since it can be completed by an action $a_{B,D}$, which does not use a suspected action and ends in state D, which has a lower bound of zero. State B has an upper bound of one third since it can be completed by an unsuspected action $a_{B,C}$ to state C, which has both upper and lower bounds of one-third probability of being abnormal.

Once we have recursively built up bounds on the probability of a suffix being abnormal, we can use these bounds to guide forward search for a plan that achieves the target probability T. Consider a plan starting with action $a_{I,A}$. Action $a_{I,A}$ was part of the plan that was observed to be abnormal. If we add $a_{I,A}$ to a partial plan, it must add probability to the chance of failure, as it is a candidate itself. After $a_{I,A}$, the system would be in state A. A plan could be completed through D. Action $a_{A,D}$ itself has zero probability of being abnormal, and using our heuristic bound, we know a completion through D must add exactly zero probability of being abnormal. Alternatively, from node A, a plan could also be completed through node C.

Action $a_{A,C}$ immediately adds probability of failure $p_t(a_{A,C})$ to our plan, and using our heuristic bound, we know that the completion through C must add exactly one third more probability of being abnormal to a plan. The precomputed heuristic, therefore, allows us to predict abnormality probability for any plan completion. The lower bound of the total plan is one third. This comes from one third from $a_{I,A}$ plus 0 from the completion $a_{A,D}$, $a_{D,G}$, The upper bound is three thirds, which is equal to the sum of one third from $a_{I,A}$ plus one third from $a_{A,C}$ and one third from $a_{C,G}$. If we complete this plan through $[a_{A,C}, a_{C,G}]$, the total plan will fail with probability 1. Note that this path is the same as the path originally taken. We already know that it fails, so it adds no new knowledge under the persistent fault assumption. If we complete this plan through the suffix $[a_{A,D}, a_{D,G}]$, the failure probability of the total plan will be one third, which is closer to $T = 0.5$. If it fails, then we will have learned that $a_{I,A}$ was the failed action. Note that there is no guarantee that a plan exists for an arbitrary value strictly between the bounds.

The general algorithm for computing heuristic bounds is as follows: The bounds are calculated recursively starting from all goal states. A goal state has an empty set of suffix plans $P_{G\to G} = \emptyset$; therefore, we set the lower bound $L_G = 0$ and upper bound $U_G = 0$. Each new state S_m calculates its bounds based on the bounds of all possible successor states $\mathcal{S}_{succ(S_m)}$ and on the failure probability of the connecting action a that causes the transition from S_m to S_n. A successor state S_n of S_m is any state that can be reached in a single step starting from S_m. In the single-fault case, if the faults applicable to the suffix and prefix are disjoint $H_{p_I\to S_n} \cap H_{a_{S_n,S_m}} = \emptyset$, we can simply add the failure probability of the action to the upper bound U_{S_m} on the failure probability of the suffix. We can update the lower bound L_{S_m} the same way, i.e.,

$$L_{S_m} = \min_{a\in\mathbb{A}(S_m)} \left(p_t\left(AB(a)\right) + L_{succ(S_m,a)}\right)$$
$$U_{S_m} = \max_{a\in\mathbb{A}(S_m)} \left(p_t\left(AB(a)\right) + U_{succ(S_m,a)}\right). \quad (20)$$

For actual implementation, we update the L_S and U_S values by running a pair of individual uniform cost searches: one for the L_S value and one for the U_S value. Uniform cost search is the form of BFS whose only ranking function is the accumulated cost on the best path from the initial node to the current node S. For updating the L_S value, the best path is the shortest path found so far, and for updating the U_S value, the best path is the longest no-loop path explored so far.

After finding the lower bound L_S and upper bound U_S values on $p_t(AB(P))$ for each search node S (plan P visits S), we can bound the ambiguity $v = |p_t(AB(P)) - T|$ by

$$\hat{v} = \begin{cases} 0, & \text{if } U_S \geq T \geq L_S \\ L_S - T, & \text{if } L_S > T \\ T - U_S, & \text{if } U_S < T. \end{cases} \quad (21)$$

As mentioned earlier, the weighted objective function f (19) combines the measure of resulting ambiguity (denoted $f_{\mathrm{DXambiguity}}$) with the plan finishing time estimate t (denoted $f_{\mathrm{ProductionTime}}$) to guide the search toward an optimal plan. The ambiguity objective $f_{\mathrm{DXambiguity}}$ is defined using the estimate $\hat{v}$. To make the search more efficient and find plans with better quality, we use an ordered sequence of rules to break ties among search nodes.

1) Favor plans with a smaller f (19).
2) Favor plans with a smaller $f_{\mathrm{DXambiguity}}$ value. Thus, prefer to have a more informative plan over a higher productivity plan.
3) Favor plans with a lower $\min(|L_S - T|, |U_S - T|)$ value. The intuition is that if there are two nodes with the same v value, then we prefer the one that we know there is one plan that is closer to achieving T. Note that L_S and U_S represent the failure probability of two real plans that are found during the uniform-cost search process.
4) Favor plans with a lower U_S value. The intuition is that if everything else is equal, we prefer plans with a lower upper bound on the failure probability so that we are more likely to select a successful plan at the end.
5) Favor plans with a lower L_S value. The intuition is the same as for the previous ranking function.
6) Favor plans with a lower t value. In essence, this is the most important measurement for productivity plans. By putting it lower than DX-related ranking functions, we favor informative over productive plans.
7) Favor plans with smaller total duration. Plans with smaller total execution times are less costly to execute and also less likely to fail.
8) Favor plans with smaller plan duration so far. This is a standard tie breaker in BFS for better search efficiency.

Improving Efficiency Through Search Space Pruning: Besides ranking functions, the upper and lower bounds on the total plan failure probability can be used to prune dominated nodes (i.e., nodes that are guaranteed to be less informative than some other visited nodes). Consider a search space with lower and upper bounds that do not straddle the target value T. If the bounds are calculated exactly by dynamic programming, then the best possible plan in this search space will be on one of the two boundaries that lies closer to the target value T. Let L_{S_n} and U_{S_n} be the lower and upper bounds of the search space $p_{I\to S_n}$, respectively. Then, let $V_{p_{I\to S_n}}$ represent a guaranteed upper bound on the closeness of a total plan $p_{I\to G}$ to T starting with the partial plan $p_{I\to S_n}$ as prefix, i.e.,

$$V_{p_{I\to S_n}} = \min\left(\left|L_{S_n} - T\left(p_{I\to S_n}\right)\right|, \left|U_{S_n} - T\left(p_{I\to S_n}\right)\right|\right) \quad (22)$$

is the closeness of the best plan to T, where $T(p_{I\to S_n})$ is the adjusted target value for the partial plan $p_{I\to S_n}$. It dominates every plan $p_{I\to S_j}$ such that

$$\bigwedge_{n\in\{i,j\}} \neg\left(L_{S_n} \leq T \leq U_{S_n}\right) \wedge \left(V_{p_{I\to S_i}} < V_{p_{I\to S_j}}\right) \Rightarrow prune(S_j). \quad (23)$$

VII. Choosing a Diagnosis Policy

A system may run for hundreds of thousands of sheets before a fault occurs, but when the system fails, it must immediately respond in order to resume full production. This response

typically takes the form of diagnosis to gather information to isolate the fault followed by repair to resolve the fault. We consider three diagnosis policies: passive diagnosis π_{pass}, explicit diagnosis π_{exp}, and pervasive diagnosis π_{perv}. In passive diagnosis, the planner is not influenced by the diagnosis engine at all, that is, plans are solely optimized for productivity. In the case of explicit diagnosis, the planner solely focuses on the needs of the diagnosis engine and, thus, creates plans that maximize information gain with regard to the fault hypotheses. In general, these plans need not be production plans. In pervasive diagnosis, the plan is to optimize for both productivity and information gain.

In order to compare the three approaches, we use a simple cost model of expected unrealized production. The expected production loss is due to the effort of isolating the faulty component (diagnosis costs) and exchanging this component (repair costs). The cost in this model represents the expected total amount of lost production due to the fault, called expected response cost $C^{\pi}(t)$, i.e.,

$$C^{\pi}(t) = c_d^{\pi}(t) + c_r(\Pr_t) \tag{24}$$

where $c_d^{\pi}(t)$ is the diagnosis costs of policy π as a function of the time t spent on diagnosis, and $c_r(\Pr_t)$ is the expected repair cost as a function of the current belief state represented as a probability distribution $\Pr_t$.

We assume that we choose some diagnosis policy and spend time t on diagnosis. At the end of time t, the system will have beliefs about the condition of its components $\Pr_t$. While the system could continue diagnosis until $\Pr_t$ unambiguously assigns the failure probability to a single component, it may be cheaper to instantly repair all currently suspected components. Diagnosis is still valuable, however, as repairs will typically cost less when $\Pr_t$ is more specific about the nature of the fault.

For a given diagnosis policy π, we can minimize the response cost by choosing the diagnosis time t that minimizes the response cost, i.e.,

$$C^{\pi *} = \min_t C^{\pi}(t) \tag{25}$$

$$= \min_t \left[c_d^{\pi}(t) + c_r(\Pr_t)\right]. \tag{26}$$

To better understand the tradeoffs between passive, explicit, and pervasive diagnosis, consider a simplified opportunity cost model. In our model, we assume that, under normal conditions, the machine produces output at its nominal rate r_{nom} and that diagnosis efforts begin once some abnormal outcome is observed (i.e., a paper jam, dog ear, etc.). Under our diagnosability assumption, we assume that we will immediately be able to detect the system failure and that our only goal is to isolate and repair the problem. For simplicity, we assume single intermittent catastrophic faults that cause the system to produce faulty output with probability f_{nom}. Therefore, if we continue regular production (without any adaptation to a detected fault), the system might still produce output with a success probability of $1 - f_{\text{nom}}$. This can be due to the nature of the fault intermittency or due to the internal multiway redundancy. However, at the moment a first failure is detected, the belief state is a uniform distribution over all fault hypotheses.

With the passive diagnosis policy, the machine continues to produce output without any adaptation to the detected fault. Therefore, the cost can be estimated by

$$c_d^{\text{pass}} = t \times r_{\text{nom}} - t \times r_{\text{nom}} \times (1 - f_{\text{nom}}) \tag{27}$$

where we subtract from the potential lost output $t \times r_{\text{nom}}$ the successfully produced output $t \times r_{\text{nom}} \times (1 - f_{\text{nom}})$.

For the explicit approach, we assume that production is suspended during diagnosis (reasonable as chances of producing the correct output are negligible); thus, diagnosis cost is equal to the nominal rate of production that would have occurred, i.e.,

$$c_d^{\text{exp}} = t \times r_{\text{nom}}. \tag{28}$$

Finally, for pervasive diagnosis, we assume that the machine produces at a reduced rate $r_{\text{perv}} \leq r_{\text{nom}}$ during diagnosis. The reduced rate results from maximizing information gain while still obeying the production constraints. Similar to the passive policy, production is faulty during pervasive diagnosis. The probability of producing a faulty output increases due to the effort of increasing the information gain. As we see in a later section, the most information can be gained by executing plans with maximum uncertainty. That leads to an increased production failure probability $f_{\text{perv}} \geq f_{\text{nom}}$ during production; thus, the cost is

$$c_d^{\text{perv}} = t \times r_{\text{nom}} - t \times r_{\text{perv}} \times (1 - f_{\text{perv}}). \tag{29}$$

For all three policies, consider a simple repair cost model: The technician receives a list of suspected printer modules in decreasing order of their fault probability. He then follows a very simple procedure: step through the list, exchange the next module, and test the machine until the machine is working properly. Using this model, repair costs can be estimated by

$$c_r(\Pr_t) = c_{\text{exc}} \sum_{i=1}^{N} i \times p_t(i) \tag{30}$$

where i indicates the number of modules that have to be exchanged with probability $p_t(i)$. We assume that each exchange takes a constant amount of time t_{exc} and, thus, has a constant cost $c_{\text{exc}} = r_{\text{nom}} \times t_{\text{exc}}$.

We use the simplified opportunity cost model to compare the different diagnosis policies. To simulate the information gained at any time by a particular diagnosis policy, we choose a simple abstraction by selecting a family of distributions. Recall that we order components in a decreasing posterior probability order. Thus, we approximate $\Pr_t$ with a monotonically declining function. We assume that the "slope" of this decreasing function depends on how much information gain can be achieved per unit time. We represent this with a "slope" parameter α. Furthermore, we assume that each policy will reach a point where it cannot gain more information. That point might be different among the policies due to the amount of internal redundancy they can exploit. For example, the passive policy optimizes for efficient production and will not explore a production plan if there exists a more efficient alternative production plan. The pervasive policy, on the other hand, would explore alternative

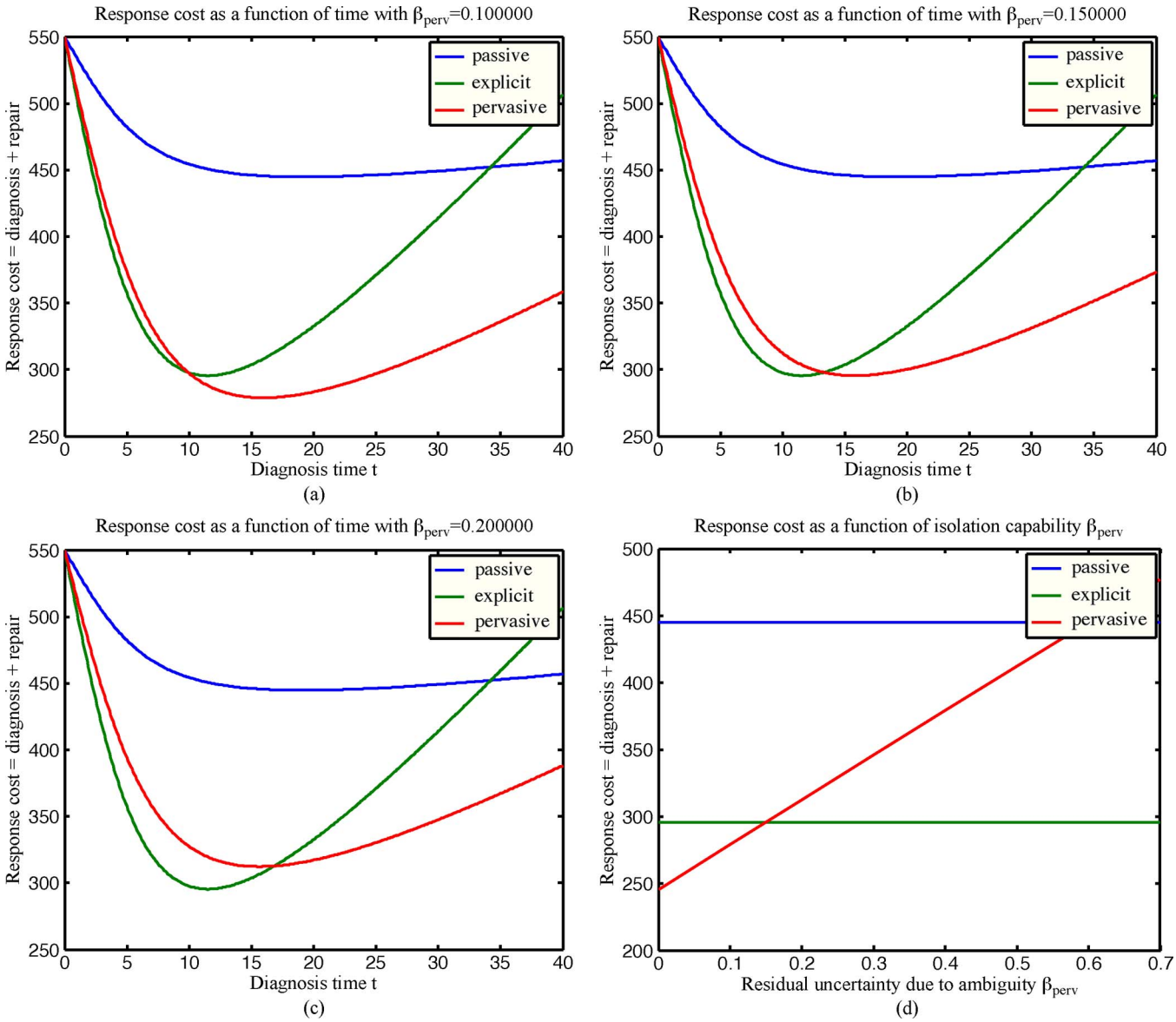

Fig. 4. Pervasive diagnosis is applicable when plans exist that create low uncertainty. (a) $\beta_{\mathrm{perv}} = 0.1$, low uncertainty, pervasive diagnosis achieves lowest cost. (b) $\beta_{\mathrm{perv}} = 0.15$, medium uncertainty, pervasive diagnosis ties with explicit. (c) $\beta_{\mathrm{perv}} = 0.2$, high uncertainty, explicit diagnosis achieves the lowest cost. (d) Lowest minimal response cost as a function of β_{perv}.

production plans to gain information. Explicit diagnosis usually gains more information since it does not have to obey the production constraint. The residual uncertainty is captured by the parameter β^{π}. Imagine that the distribution $\Pr_t$ is given by a family of the form

$$\Pr_t^{\pi} = \gamma(e^{-\alpha t} + \beta^{\pi}) \tag{31}$$

where γ is a normalizing constant to ensure that the elements of $\Pr_t$ sum to 1. Of course, this distribution does not reflect a real system, but it will allow us to see how diagnosis methods respond given their residual uncertainty β^{π}.

The response cost model allows us to understand how the cost varies with the diagnosis policy chosen and with the amount of time spent in diagnosis. We have plotted response cost over time for passive, explicit, and pervasive diagnosis policies in Fig. 4. In Fig. 4(a)–(c), we see each of the three policies as a separate curve. For short diagnosis times, we quickly start repairing the machine, but have poor information about the underlying fault. This leads to high repair cost. If we spend additional time in diagnosis, we lose output during the diagnosis, but gain information that reduces repair cost. At some point, cost for diagnosis outweighs the reduction in repair cost, and any additional time spent in diagnosis increases the response cost. Therefore, for each policy, we obtain a U-shaped curve where the bottom of the U indicates the optimal amount of diagnosis time using the policy.

Once we have determined the optimal amount of time for each policy, we must still select which of the policies to employ. The effectiveness of pervasive diagnosis depends on whether the space of production plans is sufficiently informative to isolate the fault. In Fig. 4(a)–(c), we can see that changing the residual uncertainty β_{perv} for pervasive diagnosis from 0.1 to 0.15 to 0.2 causes pervasive diagnosis to change from the

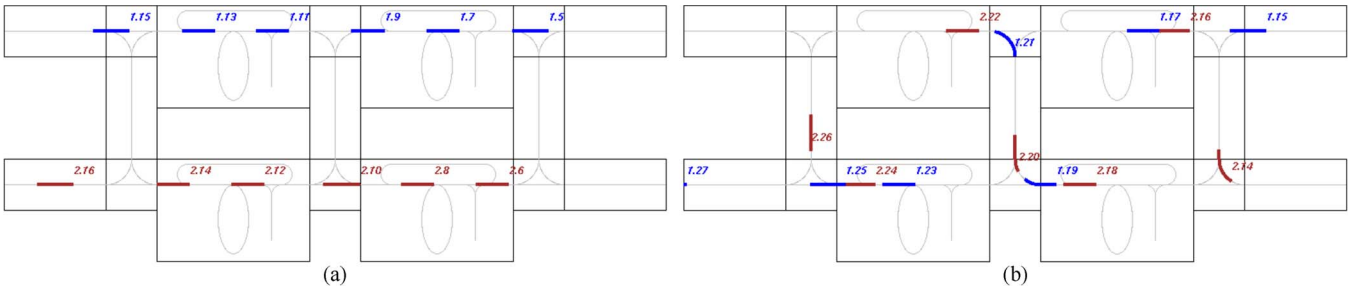

Fig. 5. Schematic of a modular printing press used in the experiments. (a) Press consists of four print engines (large rectangles) connected by controllable paper handling modules. Sheets enter on the left and exit on the right. There are 48 actions controlling feeders, paper paths, print engines, and output trays. (b) Flexibility of the architecture can be exploited to choose paper paths that use different subsets of components. A sequence of these paths can be used to isolate the fault.

diagnosis method with the lowest cost to the method with costs equal to explicit diagnosis to being more expensive than explicit diagnosis.

In Fig. 4(d), we plot the optimal cost for pervasive diagnosis as a function of residual uncertainty β_{perv} against the constant costs for explicit and passive diagnoses. We see that there is a distinct region in the lower left corner where the low cost of pervasive diagnosis dominates explicit and passive approaches. Systems in this region can use pervasive diagnosis to lower diagnosis costs without reengineering the system or adding sensors.

VIII. Experiments

To evaluate the practical benefits of pervasive diagnosis, we implemented the heuristic search. We combined it with an existing model-based planner and diagnosis engine and tested the combined system on a model of a modular digital printing press [4] or [16]. Multiple pathways allow the system to parallelize production, to use specialized print engines for specific sheets (spot color), and to reroute around failed modules. A schematic showing the paper paths in the machine appears in Fig. 5(a).

We do a test run for each possible abnormal action. The planner then receives a print job request from the queue. It generates a plan and sends it to a simulation of the printing press. The simulation models the physical dynamics of the paper moving through the system. Plans that execute on this simulation can be executed unmodified on our physical prototype machine in the laboratory. The simulation determines the outcome of the job. If the job is completed without any dog ears (bent corners) or wrinkles and deposited in the requested output tray, we say that the plan succeeded, or in the language of diagnosis, the plan was not abnormal; otherwise, the plan was abnormal.

The original plan and the outcome of executing the plan are sent to the diagnosis engine. The engine updates the fault hypothesis probabilities. When a fault occurs, the planner greedily searches for the most informative plan. Since there is a delay between submitting a plan and receiving the outcome, we plan production jobs from the job queue without optimizing for information gain until the outcome is returned. This keeps productivity high.

We evaluate the performance of passive diagnosis (only normal operation), explicit diagnosis (alternates between diagnosis and regular operation), and pervasive diagnosis (regular operation modified to obtain additional diagnostic information).

TABLE II
Pervasive Diagnosis Has the Lowest Rate of Lost Production

q=0.01	min. response cost	time t	# of exchanged modules
Passive	1010.22	214.1	2.01
Explicit	947.25	76.6	1.41
Pervasive	768.80	204.6	1.01
q=0.1	min. response cost	time t	# of exchanged modules
Passive	1055.25	35.0	2.10
Explicit	591.31	29.1	1
Pervasive	547.77	37.2	1
q=1.0	min. response cost	time t	# of exchanged modules
Passive	1012.00	7.78	2.01
Explicit	515.43	3.1	1
Pervasive	509.76	3.78	1

In our experiments, we set the exchange time for a single module to $t_{\text{exc}} = 150$ s. The nominal rate of the system is $r_{\text{nom}} = 3.1$ sheets/s. Our experiments have shown that the reduced rate of pervasive diagnosis is $r_{\text{perv}} = 1.9$ sheets/s.

Based on the introduced model, we compare the performance of passive diagnosis, explicit diagnosis, and pervasive diagnosis for three levels of fault intermittency represented by the probability q. When $q = 1$, a faulty action always causes the plan to fail. When $q = 0.01$, a faulty action only causes the plan to fail with a statistical mean of 0.01. The summary of the experimental results is in Table II. A more detailed visualization of the results is presented in Fig. 6 for the intermittency rates $q = 0.01$, $q = 0.1$, and $q = 1$. We averaged experiments over 100 runs to reduce statistical variation.

Fig. 6 shows the expected costs of repair relative to repair start time in terms of lost production in relation to a healthy machine. We plot these costs for fault intermittency rates $q = 0.01$, $q = 0.1$, and $q = 1$. The cost of repair is computed by estimating the repair time based on the current probability distribution over the fault hypothesis (see Section VII) and pricing this downtime according to the nominal machine production rate. The cost of diagnosis at time t is the accumulated production deficit in relation to a healthy machine producing at its nominal rate r_{nom} (see Section VII). The x-axis is the amount of time (relative to the first occurrence of the fault) after which one chooses to stop diagnosis and start repairing the machine. The minimum of the sum of these costs denotes the optimal point in time to switch from diagnosis to repair and gives the minimal expected total loss of production due to the fault.

In all our experiments, we found the optimal response costs of pervasive diagnosis to be below those of the other two approaches. As expected, the respective optimal durations of

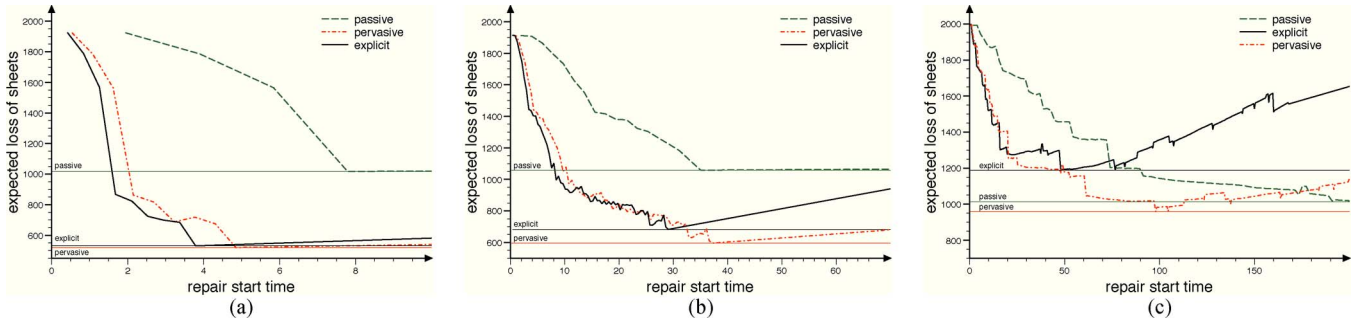

Fig. 6. Experimental results for various intermittency rates averaged over 100 runs. The curves show the response cost in expected loss of sheets as a function of diagnosis time t. For each diagnosis policy, there is a horizontal line indicating the minimal response cost. (a) $q = 1.00$. (b) $q = 0.10$. (c) $q = 0.01$.

diagnostic processes are in the order (shortest to longest) of explicit, pervasive, and passive. This corresponds to the fact that explicit diagnosis focuses solely on the diagnosis task. Therefore, explicit diagnosis is able to select plans with maximal diagnosis information gain and can isolate the faulty component in the shortest amount of time. However, due to the high production loss (production is halted), explicit diagnosis does not result in minimal response costs. Passive diagnosis has the lowest rate of lost production, but incurs the highest expected repair costs due to its lower quality diagnosis. This corresponds to the fact that the plans executed during passive diagnosis are optimized for production regardless of diagnosis needs. Pervasive diagnosis intelligently integrates diagnosis goals into production plans by using planning flexibility. Passive diagnosis works well for faults with low intermittency rates and explicit diagnosis for high intermittency; pervasive diagnosis combines the benefits of both. This leads to a lower total expected production loss in comparison to passive and explicit diagnoses.

IX. Related Work

The general mechanisms for inferring underlying causes of observations have a long history in artificial intelligence and engineering, including logic-based frameworks [17], continuous nonlinear systems [18], xerographic systems [19], and hybrid logical probabilistic diagnosis [20]. In passive diagnosis, deductive reasoning over models is used to infer the underlying condition of the system from weakly informative observations. The remote agent project [21], which is responsible for diagnosing, planning, and repairing spacecraft, is one of the most sophisticated and well-developed examples of passive diagnosis used to inform planning, search, and repair.

In active diagnosis, specific inputs or control actions are chosen to maximize diagnostic information obtained from a system. The optimal sequence of tests [5] has been calculated for static circuit domains [22]. This can be extended to sequential circuits with persistent state and dynamics through time-frame expansion methods described in [23] or [24]. The use of explicit diagnosis jobs has been suggested for model-based control systems [26].

The combination of passive diagnosis to obtain information and model-based control conditioned on the information has appeared in many domains, including automatic compensation for faulty flight controls [18], maintaining wireless sensor networks [27], and automotive engine control [28].

We are not aware of any other systems that explicitly seek to increase the diagnostic information returned by plans intended to achieve production goals.

X. Discussion

We presented an application of pervasive diagnosis to a printing domain. We believe that the technique generalizes to a wide class of production manufacturing problems in which it is important to optimize efficiency, but the cost of failure for any one job is low compared with stopping the production system to perform explicit diagnosis. Thus, the paradigm generalizes naturally to the production of grocery items, but would not generalize naturally to delivery of personal mail.

The application that we have developed illustrates a single-fault single-appearance independent fault instantiation of the pervasive diagnosis framework. To generalize the instantiation, we can make fewer assumptions by revising the representation of the hypothesis space, the belief model, and the belief update. In the most general case, i.e., multiple intermittent faults with multiple action appearance, the construction of the heuristic directly extends to an online forward heuristic computation similar to the one used in the FF planning system [29]. However, the pervasive diagnosis framework is not limited to a probability-based A^* search. Another possible instantiation could be to frame it as a satisfiability problem in which the clauses represent failed plans, and each satisfying assignment is interpreted as a valid diagnosis.

XI. Conclusion

The idea of pervasive diagnosis opens up new opportunities to efficiently exploit diagnostic information for optimizing the throughput of model-based systems. Hard-to-diagnose intermittent faults that would have required expensive production stoppages can now be addressed online during production. While pervasive diagnosis has interesting theoretical advantages, we have shown that a combination of heuristic planning and classical diagnosis can be used to create practical real-time applications, as well.

References

[1] R. Dechter, I. Meiri, and J. Pearl, "Temporal constraint networks," *Artif. Intell.*, vol. 49, no. 1–3, pp. 61–95, May 1991.
[2] R. Cervoni, A. Cesta, and A. Oddi, "Managing dynamic temporal constraint networks," in *Proc. AIPS*, 1994, pp. 13–18.

[3] D. E. Smith and D. S. Weld, "Temporal planning with mutual exclusion reasoning," in *Proc. IJCAI*, 1999, pp. 326–333.
[4] M. B. Do and W. Ruml, "Lessons learned in applying domain-independent planning to high-speed manufacturing," in *Proc. ICAPS*, 2006, pp. 370–373.
[5] R. Boumen, I. S. M. De Jong, J. M. G. Mestrom, J. M. Van De Mortel-Fronczak, and J. E. Rooda, "Integration and test sequencing for complex systems," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 39, no. 1, pp. 177–187, Jan. 2009.
[6] A. Misra, J. Sztipanovits, A. Underbrink, R. Carnes, and B. Purves, "Diagnosability of dynamical systems," in *Proc. 3rd Int. Workshop Principles Diagnosis (DX)*, Rosario, WA, 1992, pp. 239–244.
[7] A. Cimatti, C. Pecheur, and R. Cavada, "Formal verification of diagnosability via symbolic model checking," in *Proc. 18th IJCAI*, 2003, pp. 363–369.
[8] M. Krysander and M. Nyberg, "Fault isolability prediction of diagnostic models," in *Proc. 16th Int. Workshop Principles Diagnosis*, Monterey, CA, 2005.
[9] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. New York: Wiley, 1991.
[10] J. Liu, J. E. Reich, and F. Zhao, "Collaborative in-network processing for target tracking," *EURASIP, J. Appl. Signal Process.*, vol. 2003, no. 4, pp. 378–391, Mar. 2003.
[11] G. M. Hoffmann, S. L. Waslander, and C. J. Tomlin, "Mutual information methods with particle filters for mobile sensor network control," in *Proc. 45th IEEE CDC*, San Diego, CA, Dec. 2006, pp. 1019–1024.
[12] S. Ruan, Y. Zhou, F. Yu, K. R. Pattipati, P. Willett, and A. Patterson-Hine, "Dynamic multiple-fault diagnosis with imperfect tests," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 39, no. 6, pp. 1224–1236, Nov. 2009.
[13] S. Singh, A. Kodali, K. Choi, K. R. Pattipati, S. M. Namburu, S. C. Sean, D. V. Prokhorov, and L. Qiao, "Dynamic multiple fault diagnosis: Mathematical formulations and solution techniques," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 39, no. 1, pp. 160–176, Jan. 2009.
[14] R. Abreu, P. Zoeteweij, and A. van Gemund, "An evaluation of similarity coefficients for software fault localization," in *Proc. 12th IEEE PRDC*, Riverside, CA, Dec. 2006, pp. 39–46.
[15] J. de Kleer, L. Kuhn, J. Liu, B. Price, M. Do, and R. Zhou, "Continuously estimating persistent and intermittent failure probabilities," in *Proc. 7th IFAC Symp. Fault Detection, Supervision Safety Tech. Processes*, 2009, pp. 1312–1317.
[16] W. Ruml, M. B. Do, and M. Fromherz, "On-line planning and scheduling for high-speed manufacturing," in *Proc. ICAPS*, 2005, pp. 30–39.
[17] R. Reiter, "A theory of diagnosis from first principles," in *Readings in Model-Based Diagnosis*. San Mateo, CA: Morgan Kaufmann, 1992, pp. 29–48.
[18] H. E. Rauch, "Autonomous control reconfiguration," *IEEE Control Syst. Mag.*, vol. 15, no. 6, pp. 37–48, Dec. 1995.
[19] C. Zhong and P. Li, "Bayesian belief network modeling and diagnosis of xerographic systems," in *Proc. ASME Symp. Controls Imaging—IMECE*, Orlando, FL, 2000.
[20] D. Poole, "Representing diagnostic knowledge for probabilistic horn abduction," in *Proc. IJCAI*, 1991, pp. 1129–1135.
[21] N. Muscettola, P. P. Nayak, B. Pell, and B. C. Williams, "Remote agent: To boldly go where no AI system has gone before," *Artif. Intell.*, vol. 103, no. 1/2, pp. 5–47, Aug. 1998.
[22] J. de Kleer and B. C. Williams, "Diagnosing multiple faults," *Artif. Intell.*, vol. 32, no. 1, pp. 97–130, Apr. 1987.
[23] M. L. Bushnell and V. D. Agrawal, *Essentials of Electronic Testing for Digital, Memory and Mixed-Signal VLSI Circuits*. Boston, MA: Kluwer, 2000.
[24] J. de Kleer, "Troubleshooting temporal behavior in 'combinational' circuits," in *Proc. 18th Int. Workshop Principles Diagnosis*, Nashville, TN, 2007, pp. 52–58.
[25] M. F. Ali, A. Veneris, S. Safarpour, M. Abadir, R. Drechsler, and A. Smith, "Debugging sequential circuits using Boolean satisfiability," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design*, 2004, pp. 204–209.
[26] M. P. Fromherz, "Planning and scheduling reconfigurable systems with regular and diagnostic jobs," USA, Patent 7 233 405, Jun. 19, 2007.
[27] G. Provan and Y.-L. Chen, "Model-based diagnosis and control reconfiguration for discrete event systems: An integrated approach," in *Proc. 38th Conf. Decision Control*, Phoenix, AZ, 1999, pp. 1762–1768.
[28] Y.-W. Kim, G. Rizzoni, and V. Utkin, "Automotive engine diagnosis and control via nonlinear estimation," *IEEE Control Syst.*, vol. 18, no. 5, pp. 84–98, Oct. 1998.
[29] P. Haslum and H. Geffner, "Heuristic planning with time and resources," in *Proc. ECP*, 2001, pp. 121–132.

Lukas Kuhn received the computer science degree (Diploma) from the University of Munich, Munich, Germany. He is currently working toward the Doctorate degree in the Intelligent Autonomous Systems Laboratory, Technical University of Munich, Munich, under the supervision of Prof. Beetz.

He is also a Research Assistant with the Embedded Research Area, Palo Alto Research Center, Palo Alto, CA, where he works on model-based diagnosis and planning and on qualitative reasoning. His current work focuses on reasoning for planning systems with multiple intermittent faults possibly caused by hidden interactions.

Bob Price received the Ph.D. degree from The University of British Columbia, Vancouver, BC, Canada.

He is a member of the scientific staff in the Embedded Research Area, Palo Alto Research Center, Palo Alto, CA. He has served on numerous conference program committees and published articles in top international journals and conferences in areas that include accelerating reinforcement learning in multiagent systems, the use of first-order representations in Markov decision processes, decision theoretic models of recommendation set generation, Bayesian inference, and Bayesian models for tracking consumer behavior over time. His interests include machine learning, modeling, and planning under uncertainty.

Minh Do received the Ph.D. degree from Arizona State University, Tempe, in 2004.

He is currently a Research Staff Member with the Embedded Research Area, Palo Alto Research Center, Palo Alto, CA. He is an expert on planning with temporal, resources constraints, and preferences. He has been publishing widely in leading AI and planning conferences and journals.

Dr. Do was a recipient of the Best Application Paper Award at the International Conferences on Automated Planning and Scheduling in 2005. He also gave a tutorial on planning with preferences and soft constraints in the AAAI-2007 Conference. He was part of the YochanPS team that won the Distinguished Performance Award at the Fifth International Planning Competition in 2006.

Juan Liu (S'96–M'01) received the B.E. degree in electronic engineering from Tsinghua University, Beijing, China, in 1995 and the M.S. and Ph.D. degrees in electrical engineering from the University of Illinois at Urbana–Champaign, Urbana, in 1998 and 2001, respectively.

In 2001, she joined the Palo Alto Research Center, Palo Alto, CA, as a member of the research staff in the Embedded Reasoning Area. Her research interests include signal processing, statistical modeling and inference, distributed sensor networks, and applications such as intelligent transportation systems.

Dr. Liu was a recipient of the IEEE Signal Processing Society Best Young Author Paper Award in 2002.

Rong Zhou received the Ph.D. degree from Mississippi State University, Mississippi State, in 2005.

He is a member of the research staff of the Palo Alto Research Center, Palo Alto, CA, where he works in the Embedded Reasoning Area of the Intelligent Systems Laboratory. He has published widely in the field of artificial intelligence. His research interests include heuristic search and automated planning.

Dr. Zhou was a recipient of two Best Paper Awards from the International Conferences on Automated Planning and Scheduling in 2004 and 2005.

Tim Schmidt was born in Neunkirchen, Germany, on September 2, 1978. He received the Diplom-Informatiker degree from Ludwig-Maximilians-Universität, Munich, Germany, in 2006. He is currently working toward the Doctorate degree at the Technical University of Munich (TUM), Munich, under the supervision of Prof. Beetz.

From February 2007 to February 2008, he was a Research Assistant with the Software Engineering Chair, TUM. He is also currently a Research Assistant with the Palo Alto Research Center, Palo Alto, CA, working on heuristic search for parallel and probabilistic planning.

Johan de Kleer (M'94) received the Ph.D. degree in artificial intelligence from the Massachusetts Institute of Technology, Cambridge, in 1979.

He is currently a Principal Scientist with the Intelligent Systems Laboratory, Palo Alto Research Center, Palo Alto, CA. He has published widely on qualitative physics, model-based reasoning, truth maintenance systems, and knowledge representation. He has coauthored three books: *Readings in Qualitative Physics*, *Readings in Model-Based Diagnosis*, and *Building Problem Solvers*.

Dr. de Kleer was a recipient of the prestigious Computers and Thought Award at the International Joint Conference on Artificial Intelligence in 1987. He is a Fellow of the American Association of Artificial Intelligence and the Association of Computing Machinery.