Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence (2008)

Pervasive Diagnosis: The Integration of Diagnostic Goals into Production Plans

Lukas Kuhn, Bob Price, Johan de Kleer, Minh Do, Rong Zhou
Palo Alto Research Center
3333 Coyote Hill Road, Palo Alto, CA 94304 USA
lukas.kuhn@parc.com

Abstract

In model-based control, a planner uses a system description to create a plan that achieves production goals (Fikes & Nilsson 1971). The same description can be used by model-based diagnosis to infer the condition of components in a system from partially informative sensors. Prior work has demonstrated that diagnosis can be used to adapt the control of a system to changes in its components. However diagnosis must either make inferences from passive observations of production, or production must be halted to take diagnostic actions. We observe that the declarative nature of model-based control allows the planner to achieve production goals in multiple ways. This flexibility can be exploited with a novel paradigm we call *pervasive diagnosis* which produces *diagnostic production plans* that simultaneously achieve production goals while uncovering additional information about component health. We present an efficient heuristic search for these diagnostic production plans and show through experiments on a model of an industrial digital printing press that the theoretical increase in information can be realized on practical real-time systems. We obtain higher long-run productivity than a decoupled combination of planning and diagnosis.

Introduction

Artificial intelligence has long been inspired by the vision of fully autonomous systems that not only act on the larger world, but also maintain and optimize themselves. This increased autonomy, reliability and flexibility are important in domains ranging from space craft to manufacturing processes. Autonomy is the combination of two processes: diagnosis of the current condition of components in a system from weakly informative sensor readings and model-based control of system operation optimized for the current condition of the system. In an aerospace domain, flight dynamics models can be used to diagnose faults in flight control surfaces from noisy observations of flight trajectories. A model-based flight controller could then compensate for the faults by using alternative control surfaces or engine thrust to achieve the pilot's goals.

Diagnosis and model-based control are typically combined in one of two ways: 1) alternating between an *active diagnosis* phase with a model-based control phase or 2) parallel execution of a *passive diagnosis* process with model-based control. Alternating phases typically results in long periods during which regular operation must be suspended. This is particularly true when diagnosing rare intermittent faults. The combination of a passive diagnosis process with model-based control is often unsuccessful as regular operation may not sufficiently exercise the underlying system to isolate an underlying fault.

In this paper we introduce a new paradigm, *pervasive diagnosis*, in which parameters of model-based control are actively manipulated to maximize diagnostic information. Active diagnosis and model-based control can therefore occur *simultaneously* leading to higher overall throughput and reliability than a naive combination of diagnosis and regular operation. We show that *pervasive diagnosis* can be efficiently implemented by combining model-based probabilistic inference together with heuristic search. We provide an efficient general solution to the problem of creating plans that produce maximal information gain during system operation.

In the next section we examine how pervasive diagnosis compares to existing work. In subsequent sections we describe pervasive diagnosis in detail, and explain a specific instantiation of pervasive diagnosis. The implementation of this instantiation is then demonstrated on a commercial scale printing system.

Related Work

The general mechanisms for inferring underlying causes of observations have a long history in artificial intelligence and engineering including logic based frameworks (Reiter 1992), continuous non-linear systems (Rauch 1995), xerographic systems (Zhong & Li 2000), and hybrid logical probabilistic diagnosis (Poole 1991). 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 (Muscettola *et al.* 1998) 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 has been calculated for static circuit domains (de Kleer & Williams 1987). This can be extended to sequential circuits with persistent state and dynamics through time-frame expansion methods section 8.2.2 of (Bushnell & Agrawal 2000) or (de Kleer 2007). Finding explicit diagnosis tests can be done using a SAT formulation (Ali *et al.* 2004). The use of explicit diagnosis jobs has been suggested for model-based control systems (Fromherz 2007).

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 (Rauch 1995), choosing safe plans for planetary rovers (Dearden & Clancy 2002), maintaining wireless sensor networks (Provan & Chen 1999) and automotive engine control (Kim, Rizzoni, & Utkin 1998).

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

Pervasive Diagnosis

Pervasive diagnosis is a framework for simultaneously interweaving efficient production and diagnosis. The overall framework is depicted in Figure 1.

Pervasive Diagnosis (1) enables goal driven systems to actively gather information about the condition of their components without suspending production, and (2) trades off gathering information with avoiding penalties for production failures. If the penalty for failure is high, the planner uses the updated beliefs to avoid failures, but if the penalty is small the planner intentionally constructs plans with non-minimal probability of failure to increase the information gain. The framework is implemented by a loop: The planner creates a plan that achieves production goals and generates informative observations. This plan is then executed on the system. For many domains, a plan consisting of numerous actions that must be executed before a useful observation can be made. The diagnostic engine updates its beliefs to be consistent with the plan executed by the system and the observations. The diagnosis engine forwards updated beliefs to the planner. This input together with the failure penalty and the predicted future demand is used to determine what information to gather next.

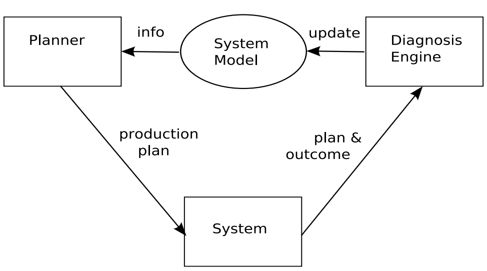

Figure 1: Framework of Pervasive Diagnosis for the Optimization of Long Run Productivity

The pervasive diagnosis framework is intended to run on real production systems during actual production so we seek efficient algorithms that can be used for planning and diagnosis in real time. An optimal solution to choosing a sequence of diagnostic plans would require explicitly reasoning about the sequences of information to be obtained. For instance, it may be the case that executing Y and Z together is more informative than executing X, even though the execution of X is better than either Y or Z alone. In this paper we implement a greedy solution to this meta-problem. We look for the most informative plan and execute it without reasoning about how information gathering affects current production efficiency or future information gathering. In many real world domains, the greedy maximization of information gain at each step is also optimal.

Representing Information

We model a system as a state machine with states $\mathbb{S}_{sys}$, actions $\mathbb{A}_{sys}$ and observables $\mathbb{O}_{sys}$. The system is controlled by a plan p consisting of a sequence of actions $a_1, a_2, \ldots, a_n$ drawn from $\mathbb{A}_{sys}$. Each action has a precondition and a postcondition. Executing an action influences how states will evolve.

While actions are generally assumed to succeed in most planning work, we admit the possibility that an action can fail. When an action fails, its postconditions are not satisfied. In keeping with conventions in the diagnosis community, we say that an action a is abnormal and write $AB(a)$.

The state of the system includes both the state of the manufacturing machine, such as the velocity of its rotors and the state of the product such as length or temperature.

Executing the actions of plan p results in an observation $o \in \mathbb{O}_{sys}$. In many systems, observations involve costly and possibly disruptive sensors. We assume a system with sensors only at the output. We abstract away from the details of the manufacturing process by considering only two outcomes: the abnormal outcome, denoted $AB(p)$, in which the plan fails to achieve its production goal and the *not* abnormal outcome, denoted $\neg AB(p)$, in which the plan does achieve the production goal.

When a plan fails to achieve its goal, the system attempts to gain information about the condition of the systems components. Information about the system is represented by the diagnosis engine's belief in various possible hypotheses. A hypothesis is a conjecture about what caused the problem. It is represented as a set of actions believed to be abnormal:

$$h = \{a | a \in \mathbb{A}_{sys},\ AB(a)\}. \tag{1}$$

The set of all possible hypotheses is denoted $\mathbb{H}_{sys}$, the power set of $\mathbb{A}_{sys}$. We distinguish one special hypothesis: the *no fault* hypothesis, h_0 under which all actions are not abnormal. We require that the set of hypotheses be mutually exclusive. If one starts with the power set of actions, this conditions will be fulfilled by construction.

The system's beliefs are represented by a probability distribution over the hypothesis space $\mathbb{H}_{sys}$, $\Pr(H)$. The beliefs are updated by a diagnosis engine from past observations using Bayes' rule to obtain a posterior distribution over the unknown hypothesis H given observation O and plan P:

$$\Pr(H|O,P) = \alpha \Pr(O|H,P)\Pr(H)$$

The probability update is addressed in more detail in a related forthcoming paper. For this paper we simply assume that we have a diagnosis engine that maintains the distribution for us.

A plan is said to be informative, if it contributes information to the diagnosis engine's beliefs. We can measure this formally as the mutual information between the system beliefs H and the plan outcome O conditioned on the plan executed p, $I(H;O|P=p)$. The mutual information is defined in terms of entropy or uncertainty implied by a probability distribution. A uniform distribution has high uncertainty and a deterministic one low uncertainty. An informative plan reduces the uncertainty of the system's beliefs. Intuitively, plans with outcomes that are hard to predict are the more informative. If we know a plan will succeed with certainty, we learn nothing by executing it. Since the entropy of a plan does not decompose additively, it turns out that it is easier to seek plans with a given failure probability T. In a forthcoming paper, we explain how to calculate the optimal plan failure probability a diagnosis engine should seek in a plan in order to maximize information. In the case of persistent faults, $T = 0.5$. In the intermittent case, the uncertainty lies in the range $0.36 \leq T \leq 0.5$.

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,

$$A_p = \bigcup_{a_i \in p} \{a_i\}. \tag{2}$$

In manufacturing systems, the failure of any production action generally results in the failure of the production plan. We say that the system is subject to *catastrophic* failures. Formally, the plan will be abnormal $AB(p)$ if any action in the plan is abnormal:

$$AB(p) \;\Leftrightarrow\; AB(a_1) \vee \cdots \vee AB(a_n) \tag{3}$$

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 H_p and is defined as:

$$H_p = \{h | a \in h,\ a \in A_p,\ h \in H_{sys}\}. \tag{4}$$

Therefore plan p will fail whenever any member of H_p is true.

$$AB(p) \;\Leftrightarrow\; h_1 \vee h_2 \vee \cdots \vee h_m \quad \text{where } h_j \in H_p \tag{5}$$

Since the hypotheses are mutually exclusive by definition, the probability of a plan failure $\Pr(AB(p))$ is defined as the sum of all probabilities of hypotheses in which plan fails:

$$\Pr(\ AB(p)\ ) = \sum_{h \in H_p} \Pr(h) \tag{6}$$

Search for Highly Diagnostic Plans

Our goal now is to find a plan which achieves production goals, but is also informative: that is, the plan is observed to have an abnormal outcome with probability T. Intentionally choosing a plan with a positive probability of failure might lower our immediate production, but the information gained allows us to maximize production over a longer time horizon. A brute force search would generate all possible sequences and filter this list to obtain the set of plans that achieve their production goals and are maximally informative. Let $\mathbb{P}_{I \to G}$ be the set of plans that achieve the production goal G. Then, the set of maximally diagnostic production plans p^{opt} is:

$$p^{opt} = \operatorname*{argmin}_{p \in \mathbb{P}_{I \to G}} \ |\ Pr(\ AB(p)\ ) - T\ |. \tag{7}$$

Unfortunately the space of plans $\mathbb{P}_{I \to G}$ is typically exponential in plan length. We can represent this plan space more compactly by considering families of plans that share structure. We can also search the space more efficiently by using a heuristic function to guide the search to explore the most promising plans (Bonet & Geffner 2001). A* (Hart, Nilsson, & Raphael 1968) performs efficient heuristic search using an open list of partial plans $p_{I \to S_1}, p_{I \to S_2}, \ldots$. These plans all start from the initial state I and go to intermediate states $S_1, S_2, \ldots$. At each step, A* expands the plan most likely to simultaneously optimize the objective and achieve the goal. A plan p can be decomposed into a prefix $p_{I \to S_n}$ which takes us to state S_n and a suffix $p_{S_n \to G}$ that continues to the goal state G. A* chooses the best partial plan $p_{I \to S_n}$ to expand using a heuristic function $f(S_n)$. The heuristic function $f(S_n)$ estimates the total path quality as $g(S_n)$, the actual quality of the prefix $p_{I \to S_n}$, plus $h(S_n)$, the *predicted* quality of the suffix $p_{S_n \to G}$.

If $f(S_n)$ never overestimates the true quality of the complete plan, then $f(S_n)$ is said to be admissible and A* is guaranteed to return an optimal plan. The underestimation causes A* to be optimistic in the face of uncertainty ensuring that promising uncertain plans are explored before committing to completed plans. The more accurate the heuristic function, the more A* will focus its search.

It is very challenging to construct an A^* search to minimize the absolute value norm $|\ Pr(\ AB(p)\ ) - T\ |$. Actions taken by the planner have an increasing effect on the total failure probability of the plan. A similar unambiguous conclusion can not be drawn for the closeness of the failure probability to T. When the probability of the plan is close to the target value, a change to the plan probability can increase or decrease the closeness of the plan to the target value. Therefore an underestimation of the total plan probability results not necessarily in a decreased closeness to the target value. In this case A^* loses its guarantee of optimality (See Figure 2).

We assume that any action can only appear once in a plan and that all faults are caused by single action failures. Given these assumptions, we can derive a particularly simple and appealing algorithm. We discuss extensions to this algorithm in later section. In the restricted case, we can decompose the search problem into cases where the heuristic is admissible and cases where it is not. Efficient A^* search is then applied when possible. Equation 8 shows a recursive decomposition of the closest probability to target value T

$$
\begin{aligned}
&\min_{p\in\mathbb{P}_{I\to G}} \; |\Pr(AB(p)) - T| \\
&= \begin{cases}
\min_{p\in\mathbb{P}_{I\to G}} \; \Pr(AB(p)) - T & \text{if } \forall_{p\in\mathbb{P}_{I\to G}} \Pr(AB(p)) \geq T \\
\max_{p\in\mathbb{P}_{I\to G}} \; T - \Pr(AB(p)) & \text{if } \forall_{p\in\mathbb{P}_{I\to G}} \Pr(AB(p)) < T \\
\min_{a\in\mathbb{A}(s)} \; \min_{p'\in\mathbb{P}_{I(a)\to G}} \; |\ \Pr(AB(p')) - T(a))\ | & \text{otherwise}
\end{cases}
\end{aligned} \tag{8}
$$

$$\text{where } I(a) = succ(I,a) \text{ and } T(a) = T - \Pr(AB(a))$$

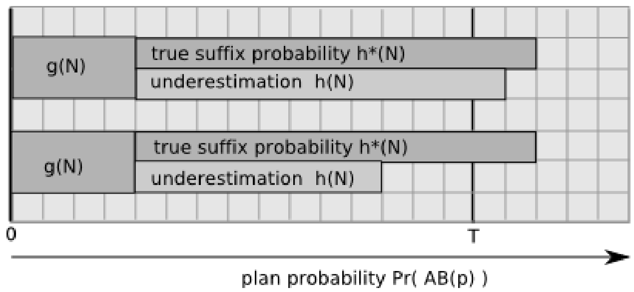

Figure 2: An underestimate of total plan probability is not guaranteed to underestimate the closeness of plan probability to target T.

that can be achieved by a plan. The plan that achieves this probability can be recovered by using auxiliary variables to track the action chosen at each step. We omit the tracking mechanism to keep the presentation clearer.

We start the search with a space of possible plans $\mathbb{P}_{I\to G}$. The value of the plan with failure probability closest to T is given by a recursion based on three cases:

1. If every plan in the space has failure probability higher than T, we simply find the plan with minimal failure probability (This can be done with A^*).
2. If every plan in the space has failure probability lower than T, we simply find the plan with maximal failure probability (This can be done with an inverted A^*).
3. Otherwise, for each possible action, we recursively search $\mathbb{P}_{I(a)\to G}$, a new subspace of plans from $I(a)$, the successor state of I under action a to the goal using the new target value $T(a) = T - \Pr(AB(a))$ adjusted by the probability of the last action chosen (Basically one step of unguided search).

To determine which case applies at any state I we can substitue lower and an upper bound on paths going through I into the expression $\forall_{p\in\mathbb{P}_{I\to G}} \Pr(AB(p)) \geq T$.

- If the plan through I with the smallest failure probability, $sp(I)$, is greater than T, then all plans through I will have greater probability than T.
- We can construct a heuristic lower bound $h^-(I)$ on the failure probability of the plan through I. If $h^-(I) > T$, then *all* plans in the space have failure probability greater than T.

Analogous arguments can be made for the plan with largest failure probability $lp(I)$ and the corresponding upper bound on the largest probability plan $h^+(I)$.

$$h^-(I) > T \Rightarrow \forall_{p\in\mathbb{P}_{I\to G}} \Pr(AB(p)) \geq T \tag{9}$$

$$h^+(I) < T \Rightarrow \forall_{p\in\mathbb{P}_{I\to G}} \Pr(AB(p)) < T \tag{10}$$

Our experiments indicate that these heuristics can significantly reduce the number of node expansions during search.

Diagnostic Plan Heuristics

In A^* search, heuristics are typically created by hand using deep insight into the domain. There are a number of methods for generating heuristics automatically. One method relies on using dynamic programming to compute actual lower and upper bounds (Culberson & Schaeffer 1998). A complete application of dynamic programming to our problem would return the optimal plan but be intractable to compute and store. A novel sparse form of dynamic programming, however provides bounds for guidance while permitting a tractable algorithm.

Consider the example in Figure 3. In this example, the graph 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*.

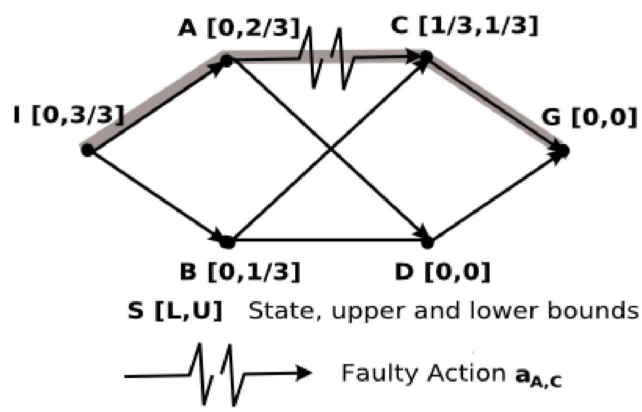

Figure 3: Machine topology places limits on what can be learned in the suffix of a plan

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 the plan resulted in an abnormal outcome. Unknown to the diagnosis engine, the abnormal outcome was caused by action $a_{A,C}$. Assume 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 would assign them equal probability (see Table 1).

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

Table 1: Probability of Hypotheses (single fault)

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 backwards from the goal state G (right side of figure). Consider action $a_{D,G}$ leading from state D to the goal state G in Figure 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 don't 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 bound for any plan ending in state D is 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 lower bound zero. State B has an upper bound of $\frac{1}{3}$ since it can be completed by an unsuspected action $a_{B,C}$ to state C which has both an upper and lower bound of $\frac{1}{3}$ probability of being abnormal.

Once we have recursively built up bounds on the probability of a suffix being abnormal, we can use these bounds with a 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 $\Pr(a_{A,C})$ to our plan and using our heuristic bound we know the completion through C must add exactly $\frac{1}{3}$ probability of being abnormal to a plan. The precomputed heuristic therefore allows us to predict total plan abnormality probability. The lower bound of the total plan is $\frac{1}{3}$. This comes from $\frac{1}{3}$ from $a_{I,A}$ plus 0 from the completion $a_{A,D}, a_{D,G}$, The upper bound is $\frac{3}{3}$ equal to the sum of $\frac{1}{3}$ from $a_{I,A}$ plus $\frac{1}{3}$ from $a_{A,C}$ and $\frac{1}{3}$ 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 it fails, so it adds no new knowledge under the persistent faults assumption. If we complete this plan through the suffix $[a_{A,D}, a_{D,G}]$ the failure probability of the total plan will be $\frac{1}{3}$ 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 any value 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 and S_n. A successor state S_n of a state S_m is any state that can be reached in a single step starting from node 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:

$$\begin{aligned} L_{S_m} &= \min_{a \in \mathbb{A}(S_m)} \Pr(AB(a)) + L_{succ(S_m,a)} \\ U_{S_m} &= \max_{a \in \mathbb{A}(S_m)} \Pr(AB(a)) + U_{succ(S_m,a)} \end{aligned} \tag{11}$$

Improving Efficiency by Search Space Pruning

In the previous section we have described an A* like search algorithm to find highly informative diagnostic plans. In the third case, the search is unguided. In this section we introduce several methods to prune nodes created during this unguided search.

The first method prunes out dominated parts of the search space. Consider a search space with a lower and upper bound that does 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, whichever one lies closest to the target value T. Let L_{S_n} and U_{S_n} be the lower and upper bound of the search space $p_{I \to S_n}$. 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. Then,

$$V_{p_{I \to S_n}} = \min(|L_{S_n} - T(p_{I \to S_n})|, |U_{S_n} - T(p_{I \to S_n})|) \tag{12}$$

will be the closeness of the best plan in the space to T. The definition uses the recursively defined adjusted target value $T(p_{I \to S_n})$. Plan $p_{I \to S_i}$ will dominate every plan $p_{I \to S_j}$ where

$$\bigwedge_{n \in \{i,j\}} \neg(L_{S_n} \le T \le U_{S_n}) \wedge V_{p_{I \to S_i}} < V_{p_{I \to S_j}} \Rightarrow prune(S_j) \tag{13}$$

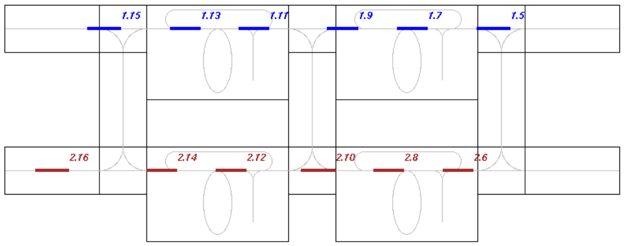

Figure 4: A schematic of a modular printing press consisting of 4 print engines (large rectangles) connected by controllable paper handling modules. Sheets enter on the left and exit on the right. There are 48 actions controlling feeders, paper paths, print engines and finisher trays.

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 (Ruml, Do, & Fromherz 2005) or (Do & Ruml 2006). Multiple pathways allow the system to parallelize production, use specialized print engines for specific sheets (spot color) and reroute around failed modules. A schematic diagram showing the paper paths in the machine appears in Figure 4.

We start a test run by uniformly choosing an action within the printing press to be abnormal. The planner then receives a job from the job queue and creates a plan that will complete the job with the target probability T. It then sends the plan 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 will execute unmodified on our physical prototype machines 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 finisher tray, we say 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 probabilities over hypotheses about which action has failed. 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. The "plan, execute, update" loop is repeated until the fault is isolated with probability $> 99.9\%$.

The results of the experiment are in Table 2. We evaluate performance 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 1/100 of the time. For each level of intermittency we compare the performance of passive diagnosis (only normal operation), explicit diagnosis (alternates diagnosis and regular operation) and pervasive diagnosis (regular operation modified for additional information). In our model, we imagine the owner of the press has a budget of 10,000 sheets which he or she would like to use as efficiently as possible. In each case, we show how many of these 10,000 sheets resulted in a successful print, how many failed and how many were used up in dedicated testing. The proportion of the 10,000 sheets that were abnormal or used for test represents the loss rate. We averaged experiments over 20 runs to reduce statistical variation.

q=0.01	Successful	Abnormal	Test	Loss Rate
Passive	9956.9	43.1	0	0.431%
Explicit	9587.7	1.0	411.3	4.123%
Pervasive	9996.8	3.2	0	0.032%
q=0.1	Successful	Abnormal	Test	Loss Rate
Passive	9636.0	364.0	0	3.640%
Explicit	9924.3	1.0	74.7	0.757%
Pervasive	9996.6	3.4	0	0.034%
q=1.0	Successful	Abnormal	Test	Loss Rate
Passive	4693.9	5306.1	0	53.061%
Explicit	9995.9	1.0	3.1	0.041%
Pervasive	9996.9	3.1	0	0.031%

Table 2: Proportion of 10,000 prints successful, failed or used for test. Here, q is intermittency rate. Pervasive diagnosis has the lowest rate of lost production.

In the persistent case, where $q = 1.0$, a faulty action causes a failure every time it is used. In the passive row of this case, we see that 53.061% of sheets are lost. In regular operation, the most efficient policy is to send two parallel streams of paper through the machine. This is shown in Figure 4. A fault in one of these paths will cause 1/2 of sheets to fail. The planner cannot isolate which action in this path fails: given only a production goal, there is no reason for the planner to consider other plans.

In pervasive diagnosis, the plan is biased to have an outcome probability close to the target T. As shown in Figure 5, the bias can create paths capable of isolating faults in specific actions. In each case, the pervasive diagnosis paradigm results in the lowest loss rate as it is able to produce useful output while performing diagnosis. On difficult intermittent problems the loss rate is an order of magnitude better than passive diagnosis and two orders of magnitude better than explicit diagnosis. We do not report timing here due to space limitations, but the algorithm runs in real time and is able to keep the job queue full.

Discussion

In the body of this paper we presented an application of pervasive diagnosis to a printing domain. We believe 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 to 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

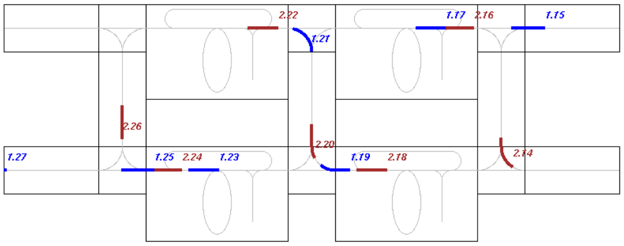

Figure 5: The flexibility of the architecture can be exploited to choose paper paths that use different subsets of components. A sequence of these paths can be used to isolate the fault.

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 reduce the set of assumptions by incorporating a different representation of the hypotheses space, the belief model and the belief update. In the most general case, 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 (Hoffmann & Nebel 2001). Due to the page limitation we omit a detailed description. However, the idea of pervasive diagnosis is not limited to a probability based A^* search. Another possibility of instantiation could be an SAT-solver approach in which the clauses represent failed plans and each satisfying assignment is interpreted as a valid diagnosis.

Conclusions

The idea of Pervasive Diagnosis opens up new opportunities to efficiently exploit diagnostic information for the optimization of the throughput of model-based systems. Hard to diagnose intermittent faults which would have required expensive production stoppages can now be addressed on line 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

Ali, M. F.; Veneris, A.; Safarpour, S.; Abadir, M.; Drechsler, R.; and Smith, A. 2004. Debugging sequential circuits using boolean satisfiability. In *Proceedings of the 2004 IEEE/ACM International conference on Computer-aided design*, 204–209.

Bonet, B., and Geffner, H. 2001. Planning as heuristic search. *Artificial Intelligence* 129(1–2):5–33.

Bushnell, M. L., and Agrawal, V. D. 2000. *Essentials of Electronic Testing for Digital, Memory and Mixed-Signal VLSI Circuits*. Boston: Kluwer Academic Publishers.

Culberson, J., and Schaeffer, J. 1998. Pattern databases. *Computational Intelligence* 14(3):318–334.

de Kleer, J., and Williams, B. C. 1987. Diagnosing multiple faults. *Artificial Intelligence* 32(1):97–130. Also in: *Readings in NonMonotonic Reasoning*, edited by Matthew L. Ginsberg, (Morgan Kaufmann, 1987), 280–297.

de Kleer, J. 2007. Troubleshooting temporal behavior in "combinational" circuits. In *18th International Workshop on Principles of Diagnosis*, 52–58.

Dearden, R., and Clancy, D. 2002. Particle filters for real-time fault detection in planetary rovers. In *Proceedings of the Thirteenth International Workshop on Principles of Diagnosis*, 1–6.

Do, M. B., and Ruml, W. 2006. Lessons learned in applying domain-independent planning to high-speed manufacturing. In *Proc. of ICAPS-06*.

Fikes, R., and Nilsson, N. 1971. Strips: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence* 2:189208.

Fromherz, M. P. 2007. Planning and scheduling reconfigurable systems with regular and diagnostic jobs. US Patent 7233405.

Hart, P.; Nilsson, N.; and Raphael, B. 1968. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics (SSC)* 4(2):100–107.

Hoffmann, J., and Nebel, B. 2001. The ff planning system: Fast plan generation through heuristic search. *J. Artif. Intell. Res. (JAIR)* 14:253–302.

Kim, Y.-W.; Rizzoni, G.; and Utkin, V. 1998. Automotive engine diagnosis and control via nonlinear estimation. *IEEE Control Systems* 84–98.

Muscettola, N.; Nayak, P. P.; Pell, B.; and Williams, B. C. 1998. Remote agent: To boldly go where no AI system has gone before. *Artificial Intelligence* 103(1-2):5–47.

Poole, D. 1991. Representing diagnostic knowledge for probabilistic horn abduction. In *International Joint Conference on Artificial Intelligence (IJCAI91)*, 1129–1135.

Provan, G., and Chen, Y.-L. 1999. Model-based diagnosis and control reconfiguration for discrete event systems: An integrated approach. In *Proceedings of the thirty-eighth Conference on Decision and Control*, 1762–1768.

Rauch, H. E. 1995. Autonomous control reconfiguration. *IEEE Control Systems Magazine* 15(6):37–48.

Reiter, R. 1992. A theory of diagnosis from first principles. In *Readings in Model-Based Diagnosis*, 29–48.

Ruml, W.; Do, M. B.; and Fromherz, M. P. J. 2005. On-line planning and scheduling for high-speed manufacturing. In *Proc. of ICAPS-05*, 30–39.

Zhong, C., and Li, P. 2000. Bayesian belief network modeling and diagnosis of xerographic systems. In *Proceedings of the ASME Symposium on Controls and Imaging - IMECE*.