Pervasive Model Adaptation: The Integration of Planning and Information Gathering in Dynamic Production Systems

Juan Liu, Lukas Kuhn, and Johan de Kleer
Palo Alto Research Center
3333 Coyote Hill Road, Palo Alto, CA 94304 USA
{jjliu,lkuhn,dekleer}@parc.com

Abstract

Model-based planning often presumes a static system model, while in a practice physical system may evolve or drift over time. This paper proposes the idea of pervasive model adaptation in a production system, where the model is dynamically updated using observation of production output. The core idea is the interplay between model adaptation and production planning. We seek plans which simultaneously serve the goals of achieving high productivity for production, and information gathering for model adaptation. We use a modular printing example to illustrate issues such as formulation of the information criterion and search strategy for informative plans. The idea of pervasive adaptation can be further extended to improve long term productivity in production systems.

Introduction

Automated planning has a proven record of achieving high productivity in production systems. In order to facilitate high productivity, a model-based planner requires suitable knowledge of the underlying system to generate plans that are feasible and optimal. Typically this knowledge can be found in the system model. However in practice, the system may evolve over time. Thus model adaptation is important to model-based tasks such as planning, scheduling, diagnosis, and inference. Ignoring dynamism in an evolving system is risky, as a production plan that started out feasible can become infeasible. Furthermore, keeping track of model parameters is important to system health management. Significant drifts can be precursors of component failures. Model adaptation techniques have been used in a number of applications to monitor aging components and detect incipient failures (e.g., (Peel 2008)).

Traditionally distinct communities of researchers have addressed model adaptation and planning seperately. For instance, planning work normally assumes static knowledge (deterministic or stochastic) regarding actions and the world. On the other hand, model adaptation is often done with passively obtained data or offline testing. There are synergistic effects between model adaptation and production planning that lead to higher productivity than a decoupled solution: model adaptation provides updated knowledge for production plan generation, and at the same time, production plans generate data for further adaptation.

In this paper, we propose a novel paradigm, *pervasive model adaptation*, in which production is actively manipulated to facilitate information gathering in order to better adapt models to dynamic systems. Active adaptation and production can therefore occur simultaneously, leading to higher long run productivity. As we shall see in the later sections, the goal of information collection may impose new optimality criteria on the planner, calling for new search techniques. Another significant challenge is model adaptation, an important research topic in its own right. However, we leave it out of this paper by assuming the optimal adaptation scheme is given and/or achievable. We are not addressing the design of a good model adaptation scheme, but rather we focus on optimally seeking data for model adaptation. The interplay between model adaptation and production planning and the resulting implications are the core focus in this paper.

Related Work

The idea of actively seeking informative evidence is not entirely new. The earlier work in general diagnosis engines (de Kleer and Williams 1987) calculates the optimal test sequence for static circuit diagnosis. Evidence seeking has also been explored in other statistical inference problems such as in Bayesian networks (Jordan 1998). The novelty of pervasive model adaptation is in combining plan generation with evidence seeking to learn continuous model parameters. A closely related concept is our earlier work on pervasive diagnosis (Kuhn et al. 2008a) (Kuhn et al. 2008b) (Liu et al. 2008), originally proposed for diagnosis of production systems with streamlined components or production actions. The basic idea is to generate production plans which achieve production goals while optimally collecting diagnosis information. This is advantageous: (1) from the productivity perspective, it does not have to halt production for offline testing (2) from the information gain perspective, the on-line diagnosis gathers information much faster than diagnosing from passive data. This paper extends the idea to model adaptation. Compared to the earlier pervasive diagnosis work, the key differentiating novelty of this paper is that it adapts to the dynamism of the underlying system, while

pervasive diagnosis assumes static faults. Furthermore, pervasive model adaptation addresses the learning of continuous model parameters, while pervasive diagnosis is centered on diagnosis, which is discrete and categorical in nature.

From the planning perspective, classical planning presumes complete knowledge of actions and the world. There is work on planning with incomplete information, with both (e.g., (Bonet and Geffner 2000)) and without sensory input (e.g., (Smith and Weld 1998)). These planning techniques can be used for pervasive model adaptation. In general, probabilistic planners can be used to generate production plans, maximizing their chance of success against a performance criterion. One differentiating factor of pervasive model adaptation is that the planner does not just accommodate uncertainties as a probabilistic planner does, but instead actively seeks evidence to reduce uncertainty in the adaptation stage.

The combination of passive information gathering 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 and Clancy 2002), maintaining wireless sensor networks (Provan and Chen 1999) and automotive engine control (Kim, Rizzoni, and Utkin 1998).

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

Organization of the Paper

The rest of the paper is organized as follows. In the next section we define the paradigm of pervasive model adaptation, followed by a section proposing a general architecture for the planner. In the next section we introduce a concrete example of pervasive model adaptation, using a modular printer whose modules independently introduce delays which drift over time, based on module usage. Model adaptation keeps track of the action delays. This is essential to determining the achievability of production goals. Though this example is specific and somewhat simplistic, it explains the major issues in pervasive model adaptation, and demonstrates how they can be addressed. The rest of the paper uses the printing example to explain the information criterion and heuristics guiding the search. At the end of the paper we demonstrate the proof of concept via simulations, and sum up with a conclusion.

Pervasive Model Adaptation

Pervasive Model Adaptation is a new paradigm in which production is actively manipulated to maximize the information available for model adaptation. Production and active information gathering can therefore occur *simultaneously*, leading to higher long run productivity than passive model adaptation or alternating dedicated testing with production.

Integrating information gathering with the production strategy results in *informative production*. The primary objective in *informative production* is to continue production. Under the assumption that there are various ways to achieve the production goals, informative production chooses a way that simultaneously maximizes information gathering with production. Another assumption is that the space of production plans overlap with the space of informative plans.

The literature describes different types of production such as simple production, time efficient production, cost efficient production and robust production. All of these share the primary objective of achieving production, but differ in the way they approach the goal. In simple production any strategy that achieves the production goal qualifies. In all other approaches the set of production strategies is ranked by a secondary objective function, and the best production strategy dominates. For example in time efficient production, strategies are ranked by cost, and the most cost efficient production strategy dominates. Similar to other production strategies, informative production ranks the set of plans that achieve production goals by their potential information gain, and selects the most promising strategy.

Planning in Pervasive Model Adaptation

Before diving into the planning work in more detail we illustrate the overall framework in Fig. 1. The basic task carried out the planner is to create production plans that simultaneously achieve production goals and favor information gathering for model adaptation. The extended task carried out by the adaptation engine is to estimate state parameters and provide information seeking guidance to the planner. Both the planner and the adaptation scheme operate with a common dynamic model of the machine state.

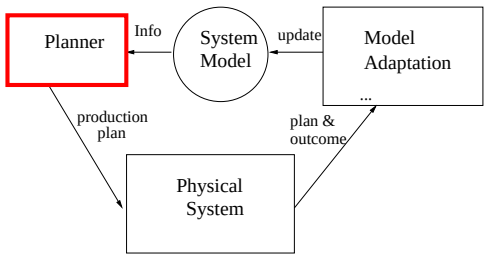

Figure 1: Overall system architecture for pervasive model adaptation.

Conceptually, planning can be thought of as the generic problem of finding an optimal plan with respect to a given cost function or optimality criterion. In traditional planning, the cost function is often assumed to have certain properties, such as being additive over plan components, decomposable into locally optimal subsets, or invariant with respect to component ordering in the plan. These structures enable efficient computation using A*-search or dynamic programming. In pervasive model adaptation, the integration of production plans and model adaptation results in *informative plans*; the cost function is the information content. A production plan is optimal if its output maximally reduces uncertainty in the system model. This information-driven cost function brings challenges to the planning problem.

In this section, we propose the general architecture for planning under pervasive model adaptation, depicted in Fig. 2. Its input is a current state (e.g., an estimate and

uncertainty), and it outputs a production plan. The planner consists of several components:

(1) Information criterion: given any production plan P, the information criterion evaluates the information content that P provides about the system. Its value depends on P, as well as the current belief about the underlying system — what we can learn from a new input depends on what we already know. This criterion is specific to pervasive model adaptation. It differs from traditional production planning, which uses well-understood criteria such as shorted production time or maximal throughput.

(2) System constraint: Production systems have internal constraints such as contingency and connectivity. Such constraints can be expressed in the planning problem. A feasible production plan is a plan from a known initial state (e.g., raw material for production) to a known goal state (e.g., finished product).

(3) Search: The core problem is to search for an optimal feasible plan with respect to the information criterion. As we shall see later, the information content of a plan P depends on all components in P, which makes the search a difficult problem with exponential complexity. The challenge is thus to find a computationally efficient search strategy which gives an informative plan. For this, we propose a best-first search strategy (A^*), which maintains a moving frontier of promising partial plans (those anticipated to produce informative plans), and expands the partial plans with highest anticipated information content until the goal state is reached. The key question is what heuristics to use to guide the search. We show how the information criterion can generate such heuristics.

The dynamic nature of production systems adds complexity to the planner. For instance, when a component ages, its behavior may deviate from the original model, and hence there is the need to track component behavior over time via adaptation. This means the planner (shown in Fig. 2) is dynamic: given a belief at time t, the planner plans for the next optimal (or close-to-optimal) plan $P^{(t)}$, which will produce an output that modifies the belief. The modified belief is fed back to the planner at time $t+1$. This is depicted as the feedback from production plan to current belief (shown in dotted lines) in the figure.

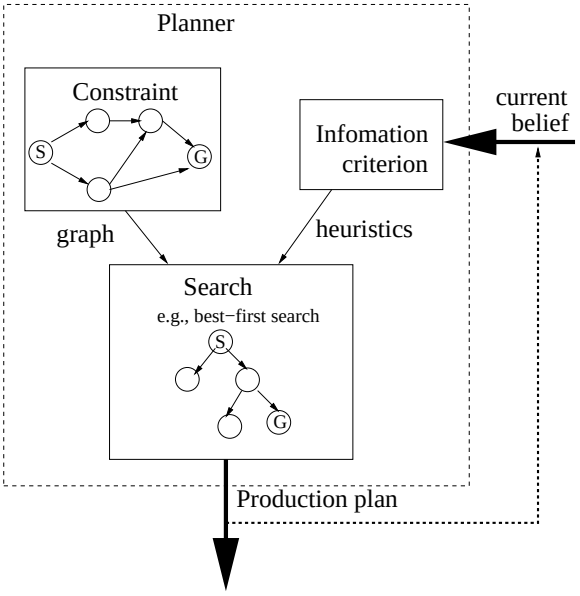

Figure 2: Planner for pervasive model adaptation

Example: Tracking Component Delay Parameters in Printing

We use a prototype modular printer as a concrete example to illustrate issues in pervasive model adaptation. Fig. 3 shows the printer. Sheets enter on the left from one of the feeders and exit on the right from a paper finisher (output), going through multiple modules in between, such as paper plan modules (little squares in the figure), inverters, or print engines. The modularized structure enables continuous printing even if some of the print engines fail or paper plan modules malfunction causing paper jams. The task of the planner is to find the sequence of actions, called a plan, which moves sheets through the system to generate the requested output.

In this prototype system, each action i takes time θ_i to handle the paper sheet, known as the action delay. The modules degrade, for example, mechanical components (spring, roller, etc) wear with usage. As a result, action delays $\{\theta_i(t)\}$ normally increase. A plan requiring a paper sheet to go from feeder to finisher within a pre-specified amount of time is feasible when the machine is new, but may become infeasible later. Hence for production systems, it is important to keep track of the action delays.

To track the drift of model parameters we need two things: (1) a dynamics model specifying how the underlying states (in this case action delays) evolve over time, and (2) an observation likelihood model relating observations to the underlying state. The dynamics model is as follows:

$$\theta_i(t) = \begin{cases} \theta_i(t-1) + \delta_i + w_i(t) & \text{for } i \in P \\ \theta_i(t-1) + w_i(t) & \text{for } i \notin P \end{cases} \quad (1)$$

The action delay $\theta_i(t)$ is influenced by two factors: (1) a wear-related increment δ_i if module i is used in plan P, and zero otherwise, and (2) a random drift $w_i(t)$. The observation is the total process time summed over all modules in the production and a random observation noise $n(t)$ as shown in the following observation model:

$$y(t) = \sum_{i \in P} \theta_i(t) + n(t). \quad (2)$$

Although this example is specific to modular printing, the idea is general and applicable to a variety of problems. For instance, the same abstract model can be used in printed circuit manufacturing, in which actions contribute to the etching process by depositing material (acid or similar) to the metallic substrate. The observed etched depth in the metallic foil is the total effect of all involved actions. More generally, the techniques presented in this paper can be directly extended to any system with linear dynamics and observation models. At a conceptual level, (Eq. 1) describes different drift patterns for modules involved and not involved, and

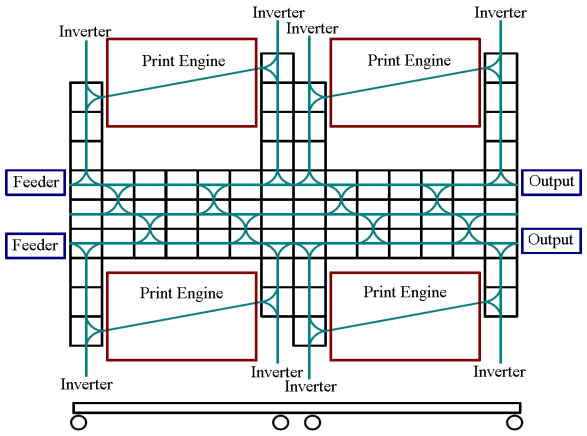

Figure 3: Model of PARC's prototype modular printer. It consists of four print engines. Sheets enters from the feeder on the left and exit from finishers (output) on the right. The little square blocks represent paper handling modules, where a paper sheet can follow one of the dark green edges. There are three main paper highways (horizontal) within the fixture.

(Eq. 2) describes an observation model that relates the observation to model parameters. As long as they are both linear, the techniques described here are applicable, although the detailed mathematical form may change. In practice, many systems can be well-represented by a linear model. Linearization techniques have been widely used to approximate real-world systems.

Information Criterion

The goal of model adaptation is to estimate the underlying model parameters (action delay $\{\theta_i(t)\}$ in the example) from the production plan output $y(t)$. The key question is: which production plan P works best to track the drift in action delay parameters? This can be addressed by examining the uncertainty in delay parameters, i.e., a production plan P is "informative" if its observation $y(t)$ reduces the uncertainty.

Information Content

For convenience in expression, we use vector notation $\Theta(t) = (\theta_1(t), \theta_2(t), \ldots, \theta_M(t))$. The models (Eq. 1, Eq. 2) can be re-written as:

$$\Theta(t) = \Theta(t-1) + \Delta(t-1) + \mathbf{w}(t) \quad (3)$$

$$y(t) = H\Theta(t) + \mathbf{n}(t) \quad (4)$$

Here H is a binary indicator vector indicating whether an action i is included in the plan P, i.e., $H_i = 1$ for all $i \in P$, and 0 otherwise. It is fully specified by the plan P. $\mathbf{\Delta}$ is the vector of increment: $\mathbf{\Delta} = (\delta_1 H_1, \delta_2 H_2, \cdots, \delta_M H_M)^T$. For simplicity, we further assume that process noise $\mathbf{w}(t)$ and observation noise $\mathbf{n}(t)$ are Gaussian. Under these assumptions, the tracking problem is relatively simple and can be tracked optimally using a Kalman filter (Kalman 1960) (Sorenson 1985).

Kalman filter updates the uncertainty in $\Theta(t)$, i.e., the covariance matrix R_θ^t, recursively over time. The recursion is as follows. Suppose at time $t-1$ we are given R_θ^{t-1}. The uncertainty grows due to the drift w; it is easy to show that the predicted covariance at time t is dilated by drift covariance, i.e., $\overline{R_\theta^t} = R_\theta^{t-1} + \Sigma_w$. Then a new observation $y(t)$ is made, bringing information to reduce the uncertainty. This updates the R_θ^t. The recursion then repeats. The Kalman filter oscillates between such uncertainty expansion and reduction. We define information content as the trace of the covariance matrix: $\mathcal{E} = trace(R_\theta^t)$. It is straightforward to show (following the standard Kalman filter algebra) that for any given production plan H, we have:

$$\mathcal{E} = \frac{||H\overline{R_\theta^t}||^2}{H\overline{R_\theta^t}H^T + \sigma_n^2}. \quad (5)$$

Note that the information depends on the predicted covariance $\overline{R_\theta^t}$. Here σ_n is the standard deviation of measurement noise n.

Another way to derive the information criterion is from information theory (Cover and Thomas 1991). The information content can be measured using the conditional entropy of Θ given observation y, which in information theory is the number of bits to fully describe Θ given the observation y. The conditional entropy is $\det(R_\Theta^t)$, which in principle is similar to (Eq. 5).

Given the information content $\mathcal{E}$, the goal of planning is to find a production plan P, or equivalently the bit vector H, to maximize the information content. The evaluation of information content is easy, but the search is difficult. In practice, there are constraints on action preconditions and module connectivity that H has to comply with. However, to gain insight, we first address a much simpler problem: if there is no constraint, what would be the best H to maximize information content?

A Simple Example

Consider the simplified case where we have only two actions A and B. Without loss of generality we assume that $\overline{R_\theta} = \begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix}$. Here ρ is the correlation coefficient, taking on values $-1 \le \rho \le 1$, and is zero if and only if A and B are independent. For this system, the planner has three choices, listed in Table 1.

Plan	H	Info.Content
A	[10]	$\mathcal{E}_A = \frac{1+\rho^2}{1+\sigma_n^2}$
B	[01]	$\mathcal{E}_B = \mathcal{E}_A$
AB	[11]	$\mathcal{E}_{AB} = \frac{2(1+\rho)^2}{2+\sigma_n^2}$

Table 1: Plan choices and information content

Notice the following:

- When $\rho = 0$, it is clear that plan AB is the best choice in terms of information content.
- When ρ is close to -1, plan A (or equivalently plan B) is more preferable than plan AB.

- The switching point of ρ is decided by σ_n^2. It is easy to specify the exact switching point. Furthermore, we can show that as $\sigma_n \to 0$, the switching point will be $\rho = 0$.

Imagine a situation when initially A and B are independent. The analysis above suggests that AB is optimal. Taking this plan, we observe the sum of delay over A and B, which will make the two modules negatively correlated. The next step will favor plan A (or equivalently B) over AB due to the negative correlation. Observing A's (or B's) individual delay will decrease the correlation between A and B, hence making ρ smaller. At some point in time, we will favor the plan AB again. The plan search will alternate between the long plan AB and individual components (A or B).

InfoCollect: a Greedy Strategy for Constructing Optimal Plans

Note the structure of the information content (Eq. 5). H is a bit vector, of values 0 or 1 depending on whether an action is involved in the production plan. The binary-valued H simplifies the computation significantly. The term in the numerator $H\overline{R_\theta}$ is the selected row-sum of the covariance matrix $\overline{R_\theta}$, summing over rows with index $i \in P$. The term in the denominator $H\overline{R_\theta}H^T$ is the grid-sum, summing over all grid points $\{(i,j),\ i,j \in P\}$.

Given the information criterion, the planner decides how to search for the optimal plan. For this task, we first ignore any network constraint on H, and propose ***InfoCollect***, a greedy strategy which constructs the plan $\mathcal{P}$ to maximize the information content (Eq. 5). The algorithm starts with an empty plan $P = \emptyset$. It incrementally decides which action to add to P by comparing the benefit of including any single module k into the plan and greedily selecting the one with maximum $\mathcal{E}$ value. The procedure repeats examining all undecided actions, until further including any single action will decrease the overall $\mathcal{E}$. This is illustrated in Alg. 1.

InfoCollect is greedy in nature. The question is how well it is compared to the true optimal plan (the "oracle"). We implemented the greedy strategy and applied it to a set of randomly generated covariance matrices. On average, among the 2^M ($M = 10$ in our simulation) possible plans, about 7% plans are better (giving larger $\mathcal{E}$) than the plan picked by ***InfoCollect***. The oracle plan gives a $\mathcal{E}$ that is roughly 55% larger than that produced by ***InfoCollect***. Overall, we consider ***InfoCollect*** as a efficient and practical way of estimating information content. The computation complexity of ***InfoCollect*** is $O(M^2)$, where M is the total number of actions. This is much more efficient than comparing all possible plans, which has complexity $O(2^M)$. The saving is apparent. Furthermore, the computation of $\mathcal{E}$ is incremental. This is due to the computational structure: the numerator is the row-sum, therefore adding a new module corresponds to adding a new row to the existing row-sum. Likewise, the denominator is a grid-sum of R, which can also be computed in an similar incremental fashion.

The ***InfoCollect*** procedure assumes a fully reachable graph and does not respect network constraints. In practice, the set of modules which can be reach from any specific module may only by a subset of all modules. Such routing constraints limit the choices of production plans. The optimal (in fact suboptimal) plan generated by ***InfoCollect*** may not be feasible.

Input: Covariance Matrix $\overline{R_\theta}$, noise variance σ_n^2
Output: (1) Optimal bit vector H and (2) the corresponding information content $\mathcal{E} = \frac{||H\overline{R_\theta}||^2}{H\overline{R_\theta}H^T + \sigma_n^2}$

begin
  Included node set: $P \leftarrow \emptyset$
  Undecided node set: $\mathcal{U} \leftarrow \mathbb{A}_{sys}$
  Initial score: $e \leftarrow 0$
  while $\mathcal{U} \neq \emptyset$ **do**
    for *all* $i \in \mathcal{U}$ **do**
      Construct P_{tmp} by including i into P;
      Compute information content e_i using P_{tmp} ;
    end
    if $\max(e_i) > e$ **then**
      Find the action i^* producing $\max(e_i)$;
      $e \leftarrow \max(e_i)$;
      Modify set: $P = P \cup i^*$, and $\mathcal{U} \leftarrow \mathcal{U} \backslash i^*$;
    else
      $\mathcal{U} \leftarrow \emptyset$
    end
  end
  Generate bit vector H from plan P ;
  Output bit vector H and information score $\mathcal{E} \leftarrow e$.
end

Algorithm 1: ***InfoCollect*** algorithm

Search for Informative Plans

The objective of pervasive adaptation is to use the adaptation engine's beliefs to influence production plans to gain additional information about the delays of individual actions.

In a planning system where actions have conditions, actions are only applicable iff the conditions satisfied. Therefore the underlying graph is not fully connected as we assumed in ***InfoCollect***. We need to find a suitable algorithm searching for a plan which maximizes the information criterion (Eq. 5) given the constraints of the action conditions. For this task, we designed a best-first-search algorithm, which keeps a frontier of best nodes to expand, until a goal state is reached. The algorithm expands and reorders nodes based on the information criterion (Eq. 5).

Best-first search requires a search node evaluation function that estimates the quality of a node relative to other search nodes in the frontier. If the evaluation function is admissible, best-first search enjoys optimality. In our case of maximizing information content, we need to estimate the information gain collected by a plan. For this task, we precompute three sets for each action and we modify the ***InfoCollect*** procedure.

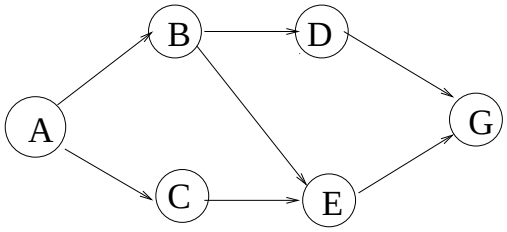

Figure 4: System used in the example.

The three sets for an action are:

$$\begin{cases} Succ_a & \text{set of successor actions} \\ Pre_a & \text{set of predecessor actions} \\ NC_a = \mathbb{A}_{sys} \backslash (Succ_a \cup Pre_a) & \text{set of actions not connected} \end{cases} \quad (6)$$

The ***InfoCollect*** procedure will give a close-to-optimal estimate of how much information can be collected given a partial plan P and the last action in that plan al. We modify simply the input bit vector H:

$$\begin{cases} H_i = 0 & \text{for } i \in \mathbb{A}_{sys} \backslash (Succ_{a\ al} \cup P) & \text{unreachable} \\ H_i = 1 & \text{for } i \in P & \text{already visited} \\ H_i = -1 & \text{for } i \in Succ_{a\ al} & \text{reachable ahead} \end{cases} \quad (7)$$

InfoCollect can be used to select which undecided actions (those with $H_i = -1$) to include in order to maximize the information content (Eq. 5). Within the ***InfoCollect*** we modify the algorithm such that if an undecided action a is selected the actions in its corresponding NC_a are exonerated (set to $H_i = 0$).

Consider the simple example graph in Fig. 4. A plan P must start at state A and reach goal state G. Initially it has two choices: B and C:

- If going to B: the visited action set is $P = \{a_{A,B}\}$. The reachable set is $Succ_a = \{a_{B,D}, a_{B,E}, a_{D,G}, a_{E,G}\}$. The non-reachable set is $\{a_{A,C}, a_{C,E}\}$. Using ***InfoCollect***, we can estimate the information content, denote as $\mathcal{E}_{AB}$
- If going to C: the visited action set is $P_2 = \{a_{A,C}\}$. The reachable set is $Succ_a = \{a_{C,E}, a_{E,G}\}$. The non-reachable set is $\{a_{A,B}, a_{B,D}, a_{B,E}, a_{D,G}\}$. Using ***InfoCollect***, we can estimate the information content, denote as $\mathcal{E}_{AC}$.

Suppose $\mathcal{E}_{AB} > \mathcal{E}_{AC}$, then going to B is a better choice. The best-first search expands the plan AB first, and maintains the frontier $\{AB, AC\}$ as a sorted list. At this point, we need to compare D and E using the same ***InfoCollect*** mechanism to compute information content $\mathcal{E}_{ABD}$ and $\mathcal{E}_{ABE}$. Now the frontier contains plan ABD, ABE, AC, sorted based on their respective information content value. The procedure repeats until a plan the goal state G is reached.

Note: although we use best-first search, we do not guarantee optimality. The reason is that the estimate generated by ***InfoCollect*** is not always admissible. It would have to always overestimate the information content to be admissible. Although the best-first search returns complete plans, ***InfoCollect*** does not obey the complete plan constraint, hence the evaluation stops including actions at a local information gain maximum, producing higher information content estimate than feasible. However, the ***InfoCollect*** algorithm itself is greedy in nature, hence it could under-estimate. This breaks the admissible requirement to guarantee optimality. Nevertheless, it provides useful heuristics to guide the search.

Experiments

To evaluate the practical benefits of pervasive model adaptation, we implemented the evaluation function ***InfoCollect***. We combined it with an existing model-based planner and adaptation engine and tested the combined system on a model of a modular digital printing press domain (see (Ruml, Do, and Fromherz 2005) and (Do and Ruml 2006)). Multiple planways allow the system to parallelize production and use specialized print engines for specific sheets (spot color). A schematic diagram showing the paper plans in the machine appears in Fig. 3.

The planner receives a job from the queue and creates a plan that will complete the job. 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 observation of the plan given the set of drifting actions. If the plan is completed without any delay and the sheet deposited in the requested finisher tray, we say the plan succeeded in time, otherwise the plan execution failed or was delayed in time. Pervasive diagnosis focuses on the case of failure ((Kuhn et al. 2008b)), which we ignore in this experiment. We focus here on plan execution delayed in time.

The original plan and the observed delay (delay might be zero) of the plan execution are sent to the adaptation engine. The engine updates the uncertainty belief model. Given the new belief the planner greedily generates new plans based on the information seeking approach used and releases those for execution. Since there is a delay between submitting a plan and receiving the observation, we plan production jobs from the queue without optimizing for information gain until the observation is returned. This keeps productivity high.

We evaluated the practical benefits of pervasive model adaptation in two experiments: In experiment 1 we compare the information gain of four different information seeking approaches. In experiment 2 we analyze the long-term productivity of four different production strategies.

Experiment 1 — Information Gain

We compare the performance of four approaches using the same model adaptation scheme, but different in their way of seeking information:

- *Regular* model adaptation (passive), which uses regular production plans (not optimized for model adaptation) to seek information, hence it is passive in its information collection strategy
- *Uniform* model adaptation, which generates production

plans with the objective to use all components of the system uniformly in usage count to seek information

- *Pervasive* model adaptation, which generates informative production plans using the approach described in the paper to seek information, hence it actively probes the parts of the system with highest uncertainty for information collection while producing at the same time.
- *Dedicated* model adaptation, which generates informative test plans using the approach described in the paper to seek information, but not producing any products. Dedicated model adaptation has fewer constraints (no production constraints) to obey than pervasive model adaptation which results in more informative plans due to the bigger plan space.

The experiment was repeated 200 times for each model adaptation approach. Within each experiment we uniformly choose six actions to drift in execution duration and performed 40 "plan, execute, observe, update" cycles.

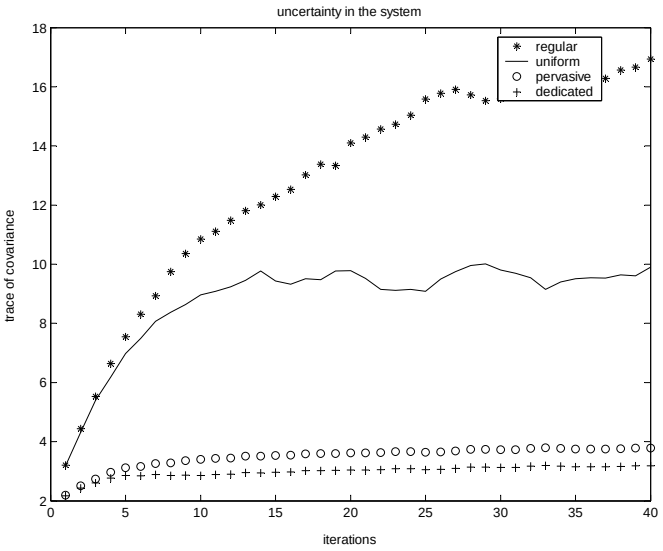

Figure 5: Comparison of regular, uniform, pervasive, and dedicated model adaptation.

The results of the experiment are in Fig. 5. The figure shows on the x-axis the number of performed "plan, execute, observe, update" iterations and on the y-axis the uncertainty as the trace of the covariance matrix ($tr(R_\theta^t)$) in unit time squared. The uncertainty is introduced by the six actions with drifting delay. Since *Regular* model adaptation is not at all optimized for adaptation it generates plans which in general do not provide enough information for the adaptation engine. We see in Fig. 5 that the uncertainty accumulates over time and *Regular* fails in limiting it. *Uniform* performs much better than *Regular* due to the partial optimization for information gathering, but finally fails in keeping the uncertainty on a steady state. In general *Uniform* will fail if the redundancy is high in the system. *Pervasive* optimizes the production plan for information gain which results in low uncertainty in the system. This can only be dominated by *Dedicated* due to the fewer constraints. Note that *Dedicated* will not produce products. Overall we can see that due to the active information gain optimization *Pervasive* and *Dedicated* perform very well in controlling the uncertainty. Finally *Dedicated* dominates all other approaches in seeking information.

Experiment 2 — Long Term Productivity

In the second experiment we compared four production strategies to assess their performance in long-term productivity. The four strategies use the same model adaptation scheme, but differ in their method of production and information seeking. Before we explain the four different strategies we define a maximum uncertainty level $maxU$ above which we switch the goal from production to information seeking. This switching avoids downtime due to failures.

- *Regular w/ dedicated.* Regular production runs until the maximum uncertainty $maxU$ is reached and then switches to dedicated information seeking.
- *Uniform w/ dedicated.* Uniform production aims to use all components of the system uniformly (by usage count). It runs until the maximum uncertainty $maxU$ is reached, and then switches to dedicated information seeking.
- *Pervasive.* Production using pervasive model adaptation, which generates informative production plans using the approach described in the paper to seek information.
- *Regular w/ pervasive.* Regular production, then switch to production using pervasive model adaptation when $maxU$ is reach.

The experiment was repeated 200 times for each production strategies. Within each experiment we uniformly chose six actions to cause to drift in execution duration and performed 30 seconds of "plan, execute, observe, update" cycles. We set the production and test rates to vary among the production strategies: regular production 3.1 $sheets/sec$, uniform production 2.0 $sheets/sec$, pervasive production 1.9 $sheets/sec$ and dedicated information seeking 2.8 $test$ $sheets/sec$. Note that dedicated information seeking does not produce valid products rather than test sheets. Regular production has the highest rate since it is optimized for throughput, the second highest rate has dedicated information seeking since it has not to obey production constraints particularly it is not always necessary to perform slow print actions. Pervasive production has the lowest rate since obeys production constraints while optimizing for information gain.

The results of the experiment are in Fig. 6. The upper figure shows the uncertainty in the system as the trace of the covariance matrix in unit time squared. We can see that all four strategies except production using *pervasive* model adaptation grow in uncertainty relatively fast until a steady state of uncertainty is reached. This is due to the fact that we switch to dedicated information seeking if the uncertainty reaches $maxU$. Since we averaged over 200 experiments the steady state of uncertainty stays under $maxU$. The distance to $maxU$ indicate the amount of information gain provided by the strategies in their production mode. *Pervasive*

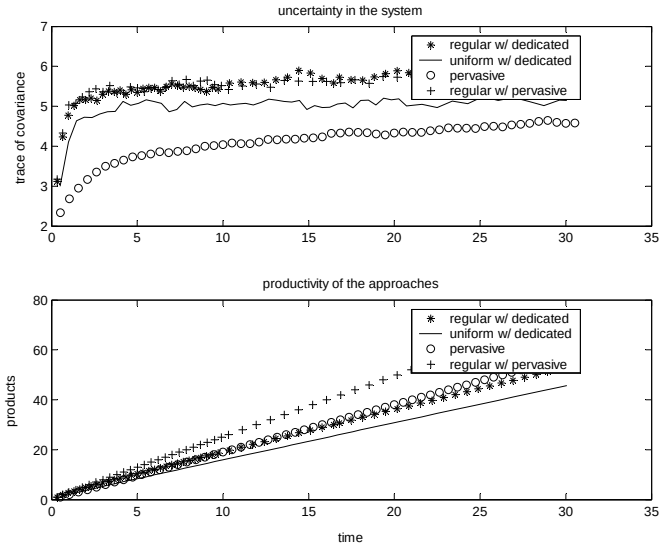

Figure 6: Comparison of different production strategies.

model adaptation stays far under $maxU$ due to its strong focus on good information seeking.

The lower figure shows the productivity of the production strategies as productions per time. The results show that *uniform w/ dedicated* has the lowest productivity. This results from neither being optimized for throughput nor information seeking. *Regular w/ dedicated* and *Pervasive* perform nearly similar. Whereas regular production has high throughput but provides low information gain, *Pervasive* has low throughput but provides high information gain. In our experiments it can be seen that effects cancel each other out. *Regular w/ pervasive* combines the strength of both worlds and performs therefore best. Overall the best strategy is to focus on throughput until the uncertainty grows to the maximum limit $maxU$ and switch than to informative production plans provided by pervasive model adaptation until the uncertainty drops.

Conclusion

The idea of Pervasive Model Adaptation opens up new opportunities to efficiently track models for the optimization of the throughput of model-based systems. Continuous tracking which would have required expensive production stoppages can now be addressed on-line during production. While pervasive model adaptation has interesting theoretical advantages, we have shown that a tight combination of heuristic planning and classical model adaptation can be used to create practical real time applications as well.

References

Bonet, B., and Geffner, H. 2000. Planning with incomplete information as heuristic search in belief space. In *Proceedings of AAAI*.

Cover, T. M., and Thomas, J. A. 1991. *Elements of Information Theory*. New York, NY: John Wiley and Sons, Inc.

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.

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 *ICAPS*, 370–373.

Jordan, M. 1998. *Learning in Graphical Models*. The MIT Press.

Kalman, R. E. 1960. A new approach to linear filtering and prediction problems. *Transactions of the ASME - Journal of Basic Engineering* 82:35–45.

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

Kuhn, L.; Price, B.; de Kleer, J.; Do, M.; and Zhou, R. 2008a. Heuristic search for target-value path problem. In *Proceedings of the 23rd AAAI Conference on Artificial Intelligence (AAAI-08)*.

Kuhn, L.; Price, B.; de Kleer, J.; Do, M.; and Zhou, R. 2008b. Pervasive diagnosis: Integration of active diagnosis into production plans. In *proceedings of AAAI*.

Liu, J.; de Kleer, J.; Kuhn, L.; Price, B.; and Zhou, R. 2008. A unified information criterion for evaluating probe and test selection. In *proceedings of Prognostics and Health Management (PHM2008)*.

Peel, L. 2008. Data driven prognostics using a kalman filter ensemble of neural network models. In *proceedings of Prognostics and Health Management (PHM2008)*.

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.

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

Smith, D., and Weld, D. 1998. Conformant graphplan. In *Proceedings of AAAI*, 889–896.

Sorenson, H. 1985. *Kalman Filtering: Theory and Application*. IEEE Press.