Constraint-Based Integration of Plan Tracking and Prognosis for Autonomous Production*

Paul Maier, Martin Sachenbacher, Thomas Rühr[1], and Lukas Kuhn[2]

[1] Technische Universität München, Department of Informatics
Boltzmanstraße 3, 85748 Garching, Germany
{maierpa,sachenba,ruehr}@in.tum.de
[2] PARC, Palo Alto, USA
Lukas.Kuhn@parc.com

Abstract. Today's complex production systems allow to simultaneously build different products following individual production plans. Such plans may fail due to component faults or unforeseen behavior, resulting in flawed products. In this paper, we propose a method to integrate diagnosis with plan assessment to prevent plan failure, and to gain diagnostic information when needed. In our setting, plans are generated from a planner before being executed on the system. If the underlying system drifts due to component faults or unforeseen behavior, plans that are ready for execution or already being executed are uncertain to succeed or fail. Therefore, our approach tracks plan execution using probabilistic hierarchical constraint automata (PHCA) models of the system. This allows to explain past system behavior, such as observed discrepancies, while at the same time it can be used to predict a plan's remaining chance of success or failure. We propose a formulation of this combined diagnosis/assessment problem as a constraint optimization problem, and present a fast solution algorithm that estimates success or failure probabilities by considering only a limited number k of system trajectories.

1 Introduction

As markets increasingly demand for highly customized and variant-rich products, the industry struggles to develop production systems that attain the necessary flexibility, but maintain cost levels comparable to mass production. A main cost driver in production is the human workforce needed for setup steps, the design of process flows, and quality assurance systems. These high labor costs can only be amortized by very large lot sizes. Therefore, to enable individualized production with its typically small lot sizes, automation must reach levels of flexibility similar to human workers. The TUM excellence cluster "Cognition for Technical Systems" (CoTeSys) [1] aims at understanding human cognition and making its performance accessible for technical systems. The goal is to develop systems that act robustly under high uncertainty, reliably handle unexpected events, and quickly adapt to changing tasks and own capabilities.

* This work is supported by DFG and CoTeSys. Preprint submitted to KI 2009.

A key technology for the realization of such systems is automated planning combined with self-diagnosis and self-assessment. These capabilities can allow the system to plan its own actions, and also react to failures and adapt the behavior to changing circumstances. From the point of view of planning, production systems are a relatively rigid environment, where the necessary steps to manufacture a product can be anticipated well ahead. However, from a diagnosis point of view, production systems are typically equipped with only few sensors, so it cannot be reliably observed whether an individual manufacturing step went indeed as planned; instead, this becomes only gradually more certain while the production plan is being executed. Therefore, in the presence of faults or other unforeseen behaviors, which become more likely in individualized production, the question arises whether plans that are ready for execution or already being executed will indeed succeed, and whether it is necessary to revise a plan or even switch to another plan.

To address this problem, we propose in this paper a model-based capability that estimates the success probability of production plans in execution. We assume that a planner provides plans given a system model. A plan is a sequence of action and start time pairs where each action is executed at the corresponding start time. Whenever the system produces an observation, it will be forwarded to a module that performs simultaneous situation assessment and plan prognostic using probabilistic hierarchical constraint automata (PHCA) models [2] of the system. We propose a formulation of this problem as a soft constraint optimization problem [3] over a window of N time steps that extends both into the past and the future, and present a fast but approximate solution method that enumerates a limited number k of most likely system trajectories. The resulting success or failure prognosis can then be used to autonomously react in different ways depending on the probability estimate; for instance, continue with plan execution, discard the plan, or augment the plan by adding observation-gathering actions to gain further information [4].

In the remainder of the paper, we first motivate the approach informally with an example from an automated metal machining process, and then present our algorithmic solution and experimental results.

2 Example: Metal Machining and Assembly

As part of the CoTeSys cognitive factory test-bed, we set up a customized and extended Flexible Manufacturing System (FMS) based on the iCim3000 from Festo AG (see figure 1b). The system consists of a conveyor transport and three stations: storage, machining (milling and turning), and assembly. We built a simplified model of this manufacturing system (see figure 1a) which consists only of the machining and the assembly station and allows to track system behavior over time, including unlikely component faults. In particular, the machining station can transition to a "cutter blunt" composite location, where abrasions are caused during operation due to a blunt cutter. This makes it very probable that the cutter breaks, leading to flawed products (see figure 1d). The assembly

station model contains a composite location which models occasional abrasions. A vibration sensor at the assembly station can detect these abrasions, yielding binary signals "abrasion occurred" and "no abrasion occurred". However, the signal is ambiguous, since the sensor cannot differentiate between the two possible causes.

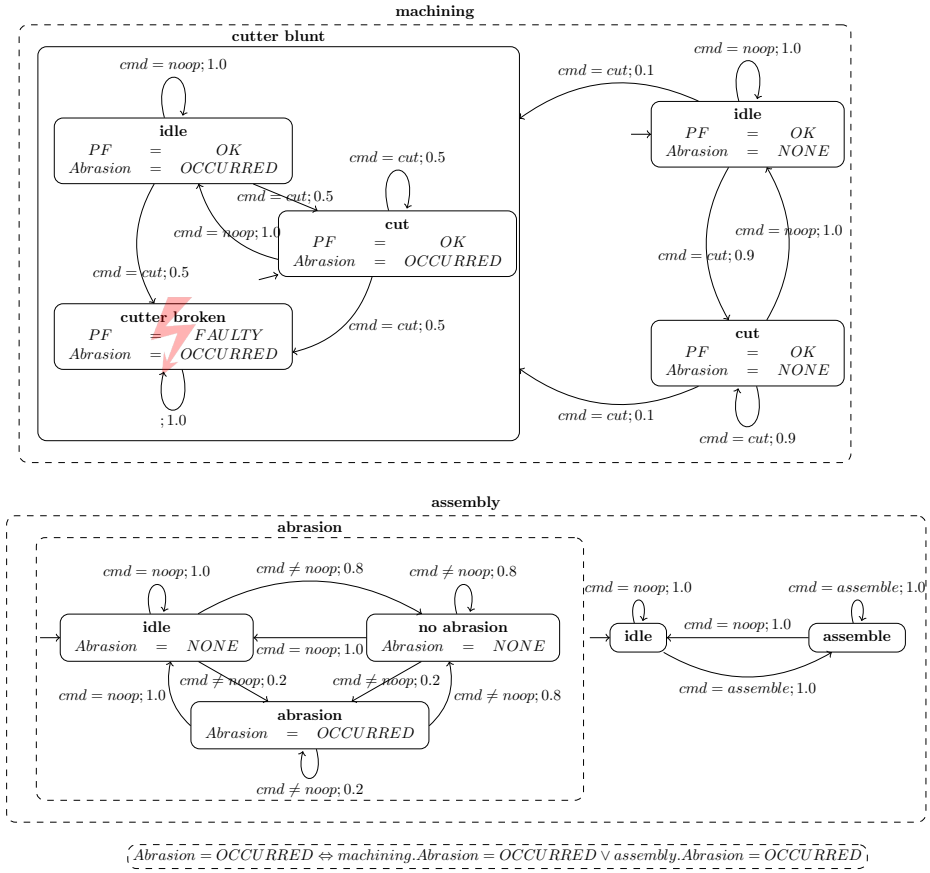
Two products are produced using a single production plan $\mathcal{P}_{prod}$: a toy maze consisting of an alloy base plate and an acrylic glass cover, and an alloy part of a robotic arm (see figure 1c). $\mathcal{P}_{prod}$ consists of these steps: (1) cut maze into base plate (one time step), (2) assemble base plate and cover (one time step), (3,4,5,6) cut robot arm part (one to four time steps). The plan takes two to six time steps (starting at $t = 0$). The plan is considered successful if both products are flawless. In our example, only a broken cutter causes the machined product to be flawed, in all other cases the production plan will succeed. Now consider the following scenario: after the second plan step (assembling the maze base plate and its cover at $t = 2$) an abrasion is observed. Due to sensor ambiguity it remains unclear whether the plan is unaffected (abrasion within assembly) or whether it might fail in the future due to a broken cutter (abrasion caused by a blunt cutter), and the question for the planner is: How likely is it that the current plan will still succeed? Our new capability allows to compute this likelihood, taking into account past observations and future plan steps.

3 Plan Assessment as Constraint Optimization over PHCA Models

Probabilistic hierarchical constraint automata (PHCA) were introduced in [2] as a compact encoding of hidden markov models (HMMs). They allow to model both probabilistic hardware behavior (such as likelihood of component failures) and complex software behavior (such as high level control programs) in a uniform way. A PHCA is a hierarchical automaton with a set of locations Σ, consisting of primitive and composite locations, a set of variables Π, and a probabilistic transition function $P_T(l_i)$. Π consists of dependent, observable and commandable variables. The state of a PHCA at time t is a set of marked locations, called a marking. The execution semantics of a PHCA is defined in terms of possible evolutions of such markings (for details see [2]).

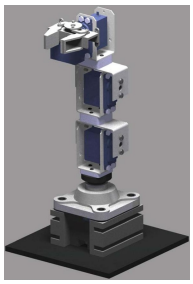
Plan assessment requires tracking of the system's plan-induced evolution; in our case, it means tracking the evolution of PHCA markings. In previous work [5], we introduced an encoding of PHCA as soft constraints and casted the problem of tracking markings as a soft constraint optimization problem [3], whose solutions correspond to the most likely sequences of markings given the available observations. The encoding is parameterized by N, which is the number of time steps considered (for a detailed description of the encoding, see [5]).

Observations made online are encoded as hard constraints specifying assignments to observable variables at time t (e.g., $Abrasion^{(2)} = OCCURRED$), and added to the constraint optimization problem.

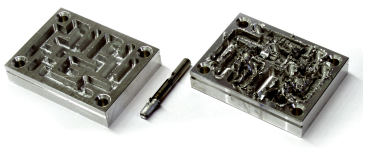


(a)

(b)



(c)



(d)

Fig. 1: (1a) Simplified PHCA of the manufacturing system. The machining and assembly station are modeled as parallel running composite locations (indicated by dashed borders). Variables appearing within a location are local to this location, i.e. *machining.cmd* refers globally to the command variable *cmd* within composite location *machining*. (1b) The hardware setup used for experimentation, showing storage, transport, robot and machining components. (1c) The robotic arm product. (1d) Effects of cutter deterioration until breakage in machining. Images *(c)* Prof. Shea TUM PE.

Plans are added analogously as assignments to commandable variables at time t; for example, $a^{(t)}_{cut}$ and $a^{(t)}_{assemble}$ are assignments $machining.cmd^{(t)} = cut \wedge assembly.cmd^{(t)} = noop$ and $assembly.cmd^{(t)} = assemble \wedge machining.cmd^{(t)} = noop$. Note that variables appear time independent in the PHCA notation (see, e.g., figure 1a), i.e. without superscript $^{(t)}$.

The plan's goal G is to produce a flawless product. We encode this informal description as a logical constraint $G \equiv \forall PF^{(t_{end})} \in RelevantFeatures(\mathcal{P}) : PF^{(t_{end})} = OK$ over product feature variables $PF^{(t)} \in \{OK, FAULTY\}$ at the end of the execution, t_{end}. $RelevantFeatures()$ is a function mapping a production plan to all product feature variables which define the product. Each system component is responsible for a product feature in the sense that if it fails, the product feature is not present ($PF^{(t)} = FAULTY$). In our example, there is only a single product feature PF, which is absent if the cutter is broken. The goal constraint for the above mentioned plan (three time steps long) is accordingly $PF^{(3)} = OK$.

We then enumerate the system's k most likely marking sequences, or trajectories, using a modified version of the soft constraint solver toolbar [6], which implements A* search with mini-bucket heuristics (a dynamic programming approach that produces search heuristics with variable strength depending on a parameter i, see [7]).

Combining Plan Tracking and Prognosis. To assess a plan's probability of success, we require not only to track past system behavior, but also to predict its evolution in the future. In principle, this could be accomplished in two separate steps: first, assess the system's state given the past behavior, and then predict its future behavior given this belief state and the plan. However, this two-step approach leads to a problem. Computing a belief state (complete set of diagnoses) is intractable, thus it must be replaced by some approximation (such as considering only k most likely diagnoses [8]). But if a plan uses a certain component intensely, then this component's failure probability is relevant for assessing this plan, even if it is very low and therefore would not appear in the approximation. In other words, the plan to be assessed determines which parts of the belief state (diagnoses) are relevant.

To address this dependency problem, we propose a method that performs diagnosis and plan assessment *simultaneously*, by framing it as a single optimization problem. The key idea is as follows: The optimization problem formulation is independent of where the present time point is within the N-step time window. We therefore choose it such that the time window covers the remaining future plan actions as well as the past behavior. Now solutions to the COP are trajectories which start in the past and end in the future. We then compute a plan's success probability by summing over system trajectories that achieve the goal. Again due to complexity reasons, we cannot do this exactly and approximate the success probability by generating only the k most probable trajectories. But since we have only a single optimization problem now, we don't have to prematurely cut off unlikely hypotheses and have only one source of error, compared

to approximating the belief state and predicting the plan's evolution based on this estimate.

Approximating the Plan Success Probability. We denote the set of all trajectories as Θ and the set of the k-best trajectories as Θ^*. A trajectory is considered successful if it entails the plan's goal constraint. We define $SUCCESS := \{\theta \in \Theta | \forall s \in \mathcal{R}_{sol}, s \downarrow_Y = \theta : F_G(s) = \text{true}\}$, where $\mathcal{R}_{sol}$ is the set of all solutions to the probabilistic constraint optimization problem, $s \downarrow_Y$ their projection on marking variables, and $F_G(s)$ is the goal constraint. $SUCCESS^*$ is the set of successful trajectories among Θ^*. The exact success probability is computed as

$$P(SUCCESS|Obs,\mathcal{P}) = \sum_{\theta \in SUCCESS} P(\theta|Obs,\mathcal{P}) = \sum_{\theta \in SUCCESS} \frac{P(\theta,Obs,\mathcal{P})}{P(Obs,\mathcal{P})} =$$

$$= \sum_{\theta \in SUCCESS} \frac{P(\theta,Obs,\mathcal{P})}{\sum_{\theta \in \Theta} P(\theta,Obs,\mathcal{P})} = \frac{\sum_{\theta \in SUCCESS} P(\theta,Obs,\mathcal{P})}{\sum_{\theta \in \Theta} P(\theta,Obs,\mathcal{P})}$$

The approximate success probability $P^*(SUCCESS^*|Obs,\mathcal{P})$ is computed the same way, only $SUCCESS$ is replaced with $SUCCESS^*$ and Θ with Θ^*. We define the error of the approximation as $E(k) := |P(SUCCESS|Obs,\mathcal{P}) - P^*(SUCCESS^*|Obs,\mathcal{P})|$. $E(k)$ converges to zero as k goes to infinity. Also, $E(k) = 0$ if $P(SUCCESS|Obs,\mathcal{P})$ is 0 or 1. However, as the example in Figure 2 shows, $E(k)$ does in general not decrease monotonically with increasing k.

Algorithm for Plan Evaluation. Plans are generated by the planner and then advanced until they are finished or new observations are available. In the latter case, the currently executed plan $\mathcal{P}$ is evaluated using our algorithm. It first computes the k most probable trajectories by solving the optimization problem over N time steps. Then, using these trajectories, it approximates the success probability of plan $\mathcal{P}$ and compares the probability against two thresholds ω_{success} and ω_{fail}. We can distinguish three cases: (1) The probability is above ω_{success}, i.e. the plan will probably succeed, (2) the probability is below ω_{fail}, i.e. the plan will probably fail or (3) the probability is in between both thresholds, which means the case cannot be decided. In the first case we simply continue execution. In the second case we have to adapt the plan to the new situation. This is done by RePlan($\mathcal{P}$, Θ^*), which modifies the future actions of $\mathcal{P}$ taking into account the diagnostic information contained in Θ^*. The third case indicates that not enough information about the system's current state is available. As a reaction, a procedure ReplanPervasiveDiagnosis($\mathcal{P}$, Θ^*) implements the recently developed *pervasive diagnosis* [4], which addresses this problem by augmenting a plan with information gathering actions.

4 Experimental Results

We ran experiments for five small variations of our example scenario, where $\mathcal{P}_{prod}$ uses the machining station zero to four times. The time window size N

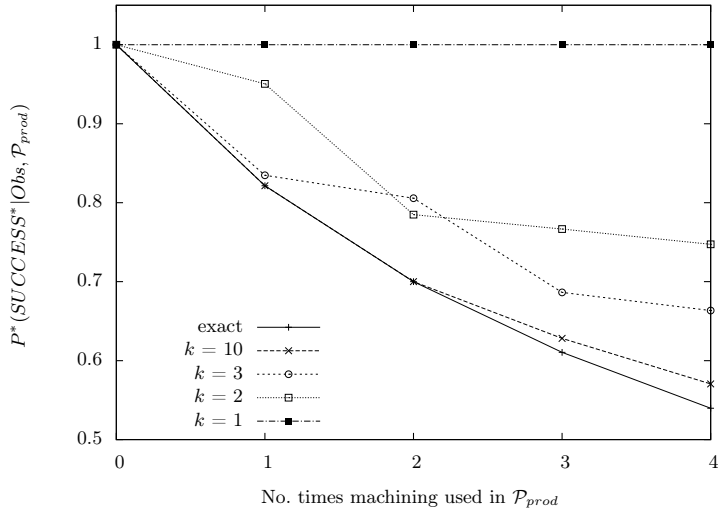
accordingly ranges from two to six, problem sizes range from 240 to 640 variables and 240 to 670 constraints. Figure 2 shows the success probabilities for different $\mathcal{P}_{prod}$ and k (top), and a table of the runtime in seconds and the peak memory consumption in megabytes (bottom) for computing success probabilities in the planning scenarios. In addition, the table ranges over different values for the mini-bucket parameter i. As expected, with increasing use of the machining station, $P^*(SUCCESS^*|Obs, \mathcal{P}_{prod})$ decreases. Also, runtime increases for larger time windows. With increasing k, $P^*(SUCCESS^*|Obs, \mathcal{P}_{prod})$ converges towards the exact solution. In our example, the approximation tends to be optimistic, which however cannot be assumed for the general case. Increasing k hardly seems to affect the runtime, especially if the mini-bucket search heuristic is strong (bigger i-values). For weaker heuristics the influence increases slightly. Memory consumption is affected much stronger by k. Here also, a weaker search heuristic means stronger influence of k.

5 Conclusion and Future Work

We presented a model-based method that combines diagnosis of past execution steps with prognosis of future execution steps of production plans, in order to allow the production system to autonomously react to failures and other unforeseen events. The method makes use of probabilistic constraint optimization to solve this combined diagnosis/prognosis problem, and preliminary results for a real-world machining scenario show it can indeed be used to guide the system away from plans that rely on suspect system components. Future work will concern the integration of the method into our overall planning/execution architecture, and its extension to multiple simultaneous plans. We are also interested in exploiting the plan diagnosis/prognosis results in order to update the underlying model, for instance, to automatically adapt to parameter drifts of components.

References

1. Beetz, M., Buss, M., Wollherr, D.: Cognitive technical systems — what is the role of artificial intelligence? In: Proc. KI-2007. (2007) 19–42
2. Williams, B.C., Chung, S., Gupta, V.: Mode estimation of model-based programs: monitoring systems with complex behavior. In: Proc. IJCAI-2001. (2001) 579–590
3. Schiex, T., Fargier, H., Verfaillie, G.: Valued constraint satisfaction problems: hard and easy problems. In: Proc. IJCAI-1995. (1995)
4. Kuhn, L., Price, B., Kleer, J.d., Do, M.B., Zhou, R.: Pervasive diagnosis: The integration of diagnostic goals into production plans. In: Proc. AAAI-2008. (2008)
5. Mikaelian, T., Williams, B.C., Sachenbacher, M.: Model-based Monitoring and Diagnosis of Systems with Software-Extended Behavior. In: Proc. AAAI-05. (2005)
6. Bouveret, S., Heras, F., Givry, S., Larrosa, J., Sanchez, M., Schiex, T.: Toolbar: a state-of-the-art platform for wcsp. www.inra.fr/mia/T/degivry/ToolBar.pdf (2004)
7. Kask, K., Dechter, R.: Mini-bucket heuristics for improved search. In: Proc. UAI-1999. (1999) 314–32
8. Kurien, J., Nayak, P.P.: Back to the future for consistency-based trajectory tracking. In: Proc. AAAI-2000. (2000) 370–377



k	i	No. times machining used in $\mathcal{P}_{prod}$ (window size N, #Variables, #Constraints)				
		0 (2, 239, 242)	1 (3, 340, 349)	2 (4, 441, 456)	3 (5, 542, 563)	4 (6, 643, 670)
1	10	< 0.1 / 1.8	0.1 / 6.8	0.1 / 19.0	(mem)	(mem)
	15	0.1 / 1.9	0.3 / 4.2	0.5 / 7.8	0.5 / 16.6	0.8 / 32.0
	20	0.1 / 1.9	0.5 / 5.2	3.7 / 20.1	6.5 / 34.5	9.5 / 50.7
2	10	< 0.1 / 2.1	0.1 / 11.9	0.2 / 38.5	(mem)	(mem)
	15	0.1 / 2.2	0.3 / 5.4	0.5 / 9.7	0.6 / 28.0	0.8 / 52.0
	20	0.1 / 2.2	0.5 / 6.4	3.7 / 21.8	6.5 / 37.2	9.5 / 55.8
3	10	< 0.1 / 2.3 (e)	0.1 / 11.9	0.2 / 40.1	(mem)	(mem)
	15	0.1 / 2.4 (e)	0.3 / 5.4	0.5 / 11.4	0.6 / 29.9	0.9 / 55.5
	20	0.1 / 2.4 (e)	0.5 / 6.4	3.7 / 23.5	6.6 / 38.3	9.5 / 57.4
4	10	(e)	0.1 12.5	0.2 / 40.1	(mem)	(mem)
	15	(e)	0.3 / 5.9	0.5 / 11.4	0.6 / 30.9	0.9 / 57.2
	20	(e)	0.5 / 6.9	3.7 / 23.5	6.6 / 39.3	9.5 / 59.1
5	10	(e)	0.1 / 13.1	0.2 / 40.7	(mem)	(mem)
	15	(e)	0.3 / 6.6	0.5 / 12.0	0.6 / 33.6	0.9 / 59.5
	20	(e)	0.5 / 7.6	3.7 / 24.0	6.6 / 42.8	9.5 / 63.9
10	10	(e)	0.1 / 14.0 (e)	0.2 / 43.4 (e)	(mem)	(mem)
	15	(e)	0.3 / 6.7 (e)	0.5 / 14.7 (e)	0.6 / 36.2	0.9 / 64.8
	20	(e)	0.6 / 7.7 (e)	3.8 / 26.6 (e)	6.6 / 45.8	9.6 / 68.9

Fig. 2: Top: Approximate success probability (y-axis) of plan $\mathcal{P}_{prod}$ against varying usage of the machining station (x-axis) after the observation of an abrasion at $t = 2$. Bottom: Runtime in seconds / peak memory consumption in megabytes. (e) indicates that the exact success probability $P(SUCCESS|Obs, \mathcal{P}_{prod})$ could be computed with this configuration. (mem) indicates that A* ran out of memory (artificial cutoff at > 1 GB, experiments were run on a Linux computer with a recent dual core 2.2 Ghz CPU with 2 GB RAM).