EFFECTIVE MAINTENANCE OF STOCHASTIC SYSTEMS VIA DYNAMIC PROGRAMMING

J. Berka, K. Macek

Department of Mathematics
Faculty of Nuclear Sciences and Physical Engineering
Czech Technical University in Prague
Trojanova 13, 120 00 Praha 2

Abstract

This paper deals with the diagnostics and maintenance of a dynamic stochastic system. A methodology for both (i) fault detection and diagnostics and (ii) efficient maintenance is introduced; the diagnostics makes use of Bayesian approach to uncertainty, while the maintenance strategy is determined by a dynamic programming algorithm. Methods for finding a suboptimal but nevertheless usable maintenance strategy with better time complexity are described. Finally, a demonstration of the methodology on an HVAC (Heating, Ventilation, Air Conditioning) system is provided.

1 Introduction

Efficient operation of systems has been examined by research for a long time. Beside optimal control, which typically assumes the system is healthy, there is wide field of diagnostics and maintenance. The purpose of diagnostics is to determine faults which are present in the system. The maintenance on the other hand makes decision about actions which repair the system, possibly to the original condition.

There is a variety of diagnostic methods; some of them focus on symptoms of abnormal behavior detection, others make also some statistical or logical inference about multiple measurements or symptoms. Some state of the art approaches are provided in Section 3. However, the maintenance is not present in the literature so much. Sometimes, the operator is assumed to decide only with respect to results of the diagnostics about probable faults. This approach is denoted as Condition Based Maintenance [6] where the actual belief on the system state determines the action, directly. There are - however - already some research efforts to schedule the maintenance in an optimal way [1].

The paper is organized as follows: Section 2 presents basic terms and used terminology, Section 3 enumerates important building blocks and shows the way how they can be put together into one algorithm. Afterwards, Section 4 introduces some other methods for finding reasonable maintenance strategies and aiming for lower computation complexity. Section 5 then describes the problem of obtaining observation and dynamic model and the loss function for a given system. In Section 6 the presented methods are used for maintenance of a HVAC system discussed in [14]. Results of different maintenance strategies composed on the same approach to system diagnostics show the importance of the maintenance question.

2 Basic Terminology and Notations

In this section we introduce basic terms of diagnostics and maintenance. We start with the definitions of stochastic dynamic system and its dynamic (sometimes called also action) and observation model. Then history is presented and next, loss function, essential for measuring of maintenance success, is described. Finally, the concept of faults and the term maintenance strategy are defined.

Definition 1 (System)**.** *Let $N_x, N_u, N_y, t_{\max} \in \mathbb{N}$, vector $\mathbf{x}^{(0)} \in \mathbb{R}^{N_x}$, conditional probability densities $f(\mathbf{x}|\mathbf{y},\mathbf{z})$, $g(\mathbf{x}|\mathbf{y},\mathbf{z})$. Then the ordered triplet $(\mathbf{x}^{(0)}, f, g)$ is called a (stochastic) **dynamic system**. The number $t_{\max}$ is **time horizon**. Vectors $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(t_{\max})} \in \mathrm{U}_x \subseteq \mathbb{R}^{N_x}$ are (hidden) **system states** and their elements are realizations of random variable specified by probability density f.*

For $\tau \in (0, t_{\max}]$, $\mathbf{U}^{(\tau-1)} = \mathbf{u}^{(\tau-1)}$, and $\mathbf{X}^{(\tau-1)} = \mathbf{x}^{(\tau-1)}$

$$\mathbf{Pr}(\mathbf{X}^{(\tau)} \leq \mathbf{x}) = \int_{-\infty}^{\mathbf{x}} f(\mathbf{x}'|\mathbf{u}^{(\tau-1)}, \mathbf{x}^{(\tau-1)})\mathrm{d}\mathbf{x}'$$

*Vector $\mathbf{x}^{(0)}$ is called the **initial state**. Vectors $\mathbf{u}^{(1)}, \ldots, \mathbf{u}^{(t_{\max})} \in \mathrm{U}_u \subseteq \mathbb{R}^{N_u}$ are called **system inputs** or **actions** and are selected by the decision maker. Vectors $\mathbf{y}^{(1)}, ..., \mathbf{y}^{t_{\max}} \in \mathbb{R}^{N_y}$ are called **system outputs** and their elements are realizations of random variable specified by probability density g.*

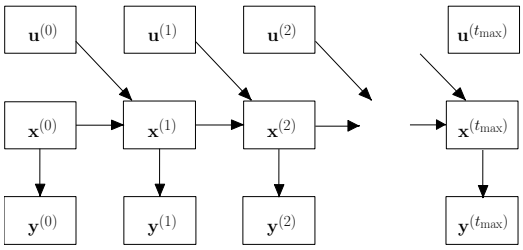

Figure 1: Diagram of a Dynamic System

For $\tau \in (0, t_{\max}]$, $\mathbf{U}^{(\tau)} = \mathbf{u}^{(\tau)}$, *and* $\mathbf{X}^{(\tau)} = \mathbf{x}^{(\tau)}$

$$\mathbf{Pr}(\mathbf{Y}^{(\tau)} \leq \mathbf{y}) = \int_{-\infty}^{\mathbf{y}} g(\mathbf{y}'|\mathbf{u}^{(\tau)}, \mathbf{x}^{(\tau)})\mathrm{d}\mathbf{y}'$$

The density functions f, g *are called* ***dynamic model*** *and* ***observation model*** *of the system.*

Note: In this paper we will expect the outputs to be conditionally independent of system inputs,

$$g(\mathbf{y}^{(t)}|\mathbf{u}^{(t)}, \mathbf{x}^{(t)}) = g(\mathbf{y}^{(t)}|\mathbf{x}^{(t)}).$$

Definition 2 (History)**.** *Let* $(\mathbf{x}^{(0)}, f, g)$ *is a dynamic system,* $t_{\max} \in \mathbb{N}$ *time horizon,* $t \in (0, t_{\max}]$*. Then the set of observed outputs and selected inputs (system output and input realizations) from initial state to time instant* t

$$H(t) = \{\mathbf{y}^{(0)}, \mathbf{u}^{(0)}, \ldots, \mathbf{y}^{(t-1)}, \mathbf{u}^{(t-1)}, \mathbf{y}^{(t)}\}$$

is called the history (in time t*).*

The system states cannot be observed directly by the decision maker, he can only compute their probabilities from the known history of inputs and outputs and dynamic and observation models of the system. Figure 1 shows the propagation of dynamic system in time.

History in time instant τ is the set of observed outputs and selected inputs from initial time $t = 0$ to time instant $t = \tau$ (without the input $\mathbf{u}^{(\tau)}$, as the history represents all the information the decision maker has in time step τ).

Definition 3 (Loss function)**.** *Let* $(\mathbf{x}^{(0)}, f, g)$ *is a dynamic system,* $U_x \in \mathbb{R}^{N_x}$ *is a set of possible system states,* $U_u \in \mathbb{R}^{N_u}$ *set of possible inputs,* $\tau \in (0, t_{\max}]$*. Then the mapping*

$$Z : \underbrace{U_x \times U_u \times \ldots \times U_x \times U_u}_{(\tau+1)-\text{times}} \to [0, \infty)$$

is called the loss function (in time instant τ*), its value*

$$Z(\mathbf{x}^{(0)}, \mathbf{u}^{(0)}, \ldots, \mathbf{x}^{(\tau)}, \mathbf{u}^{(\tau)})$$

is the loss to the moment τ*.*

Loss function evaluates the behaviour and inputs of the system and can for example represent the cost of the system life. In this case it can be divided into two parts, one evaluating costs of system states (e.g. malfunctioning motor has greater power consumption, which affects the electricity bill) and the other reflecting the costs of system inputs (repair of the motor is usually not free of charge).

The goal of system maintenance is to select such inputs, so the expected value of loss to the moment of time horizon is minimal, i. e. to minimize the expectation of the overall loss function.

When the loss function meets the following condition, it is called additive loss function:

$$\forall t \in [0, t_{\max}] \forall (\mathbf{x}^{(0)}, \ldots, \mathbf{x}^{(t)}) \forall (\mathbf{u}^{(0)}, \ldots, \mathbf{u}^{(t)}) :$$

$$Z(\mathbf{x}^{(0)}, \mathbf{u}^{(0)}, \ldots, \mathbf{x}^{(t)}, \mathbf{u}^{(t)}) = \sum_{\tau=0}^{t} z(\mathbf{x}^{(\tau)}, \mathbf{u}^{(\tau)})$$

for some function $z : U_x \times U_u \to [0, \infty)$ (named partial loss function).

The system state is not known to the decision maker, but we take the assumption, that the loss can be computed after the run of the system ends.

System states are vectors from any subset of $\mathbb{R}^{N_x}$. However, it can be very useful to divide them into several subsets and consider only these subsets instead of the states. Imagine that the decision maker has the task to diagnose and maintain an HVAC system. The system state is a long vector of temperatures in different parts of the system, air velocities, humidities, thermostat settings and so on. This might be too much information for reasonable computational complexity of the decision making. Furthermore, the conclusions from this diagnostics will be sufficient, typically: the decision maker is provided with the information which part of the system is malfunctioning and why. Let us now provide a correct definition of faults:

Definition 4. *Let* $\mathrm{U}_x \subseteq \mathbb{R}^{N_x}$ *be the space of all possible system state realizations,* $\mathcal{F} = \{\mathrm{F}_i \subset \mathrm{U}_x | i \in \{0, \dots, n\}\}$ *finite set of subsets of this space, where* F_0 *is disjunct with all other* F_i*. Then we call sets* F_i *faults and specially* F_0 *faultless state of the system.*

In this paper all faults from the set $\mathcal{F}$ will be disjunct for the purpose of simplicity in work with probabilities in Theorems 1 and 2. For systems with non-disjunct faults one can easily make them disjunct: e.g. in a cooling valve system with possible leaking and stuck faults, the disjunct fault sets (with faultless state) are all right, leaking and not stuck, not leaking and stuck, leaking and stuck. The set $\mathcal{F}$ contains also a "zero" fault, i.e. when the system works well.

As was said above, the maintenance deals with the choice of inputs. Natural condition on the maintenance is that it should reflect, what the decision maker knows (or, to be more precise, thinks) of the system state. As the system run lasts only to a set time horizon, the input choice can also depend on time.

Definition 5 (Maintenance Strategy)**.** *Let* $(\mathbf{x}^{(0)}, f, g)$ *be a dynamic system,* $t_{\max}$ *time horizon,* $\mathcal{F}$ *set of faults. The mapping*

$$S : \{0, \dots, t_{\max}\} \times [0, 1]^{|\mathcal{F}|} \to \mathrm{U}_u$$

is called a maintenance strategy.

As maintenance strategy we understand a set of rules, what input should be chosen at a given time instant $t \in [0, t_{\max}]$ under given probabilities $\mathbf{Pr}(\mathbf{x}^{(t)} \in \mathrm{F}_i | H(t))$. With a given strategy, after system run the overall loss can be computed. Intuitively, the better strategy, the less should the result be. However, as outputs and states are realizations of random vectors, with a given strategy each run can end up with completely different loss. Therefore, optimal strategy is the one minimizing the expected value of the loss with respect to state and output realizations.

The whole set of probabilities $B(t) = \{\mathbf{Pr}(\mathbf{x}^{(t)} \in \mathrm{F}_i | H(t)) | i = 0, \dots, n\}$ will be further called a (Bayesian) belief[1] of system state in time t. It is worth mentioning that $B(t) \subset [0, 1]^{|\mathcal{F}|}$ and of course

$$\forall t \in [0, t_{\max}] \sum_{i=0}^{n} \mathbf{Pr}(\mathbf{x}^{(t)} \in \mathrm{F}_i | H(t)),$$

as it is certain, then the system is in some fault (or faultless state).

3 Proposed Methods

This section summarizes several useful tools leading to efficient maintenance and diagnostics. First, symptoms are mentioned. A symptom expresses that the system seems to be in a faulty, abnormal or other considerable situation. However, the conclusions can be drawn but in the second step, namely by filtration which helps to transfer symptoms to posterior probabilities of faults. The third useful tool is the prediction which provides the possibility to forecast future evolution of the system. Finally, dynamic programming can be employed to combine all previous tools to find optimal maintenance schedule.

All of these tools have some assumptions or requirements on data and their processing. Sometimes, these data are not available and they are hard to be simulated. All these requirements are discussed always at the end of the description of the particular tool.

3.1 Symptoms

Making decisions directly based only on observed variables might be difficult because of their high dimension. So called curse of dimensionality is a serious issue. It has two alternatives: Optimization curse of dimensionality means some problems are hard to be solved for higher dimensions, while

[1] The term belief denotes the probabilities how much particular states are considered.

the statistical curse of dimensionality means higher dimensions require huge amount of training data [13]. As it will be shown later, both curses of dimensionality in the problem of efficient diagnostics and maintenance can occur.

Therefore, it is reasonable to extract some informative characteristics from the data. We will call them symptoms. Symptoms are usually binary, hence we can interpret them as propositions about recently observed data. Let us provide a formal and generic definition of a symptom:

Definition 6. *Let* $H = \{(\mathbf{Y}^{(\tau)}, \mathbf{U}^{(\tau)})\}_{\tau=1}^{t}$ *be the history of given system. Let* (Ω, R, P) *be the probability space corresponding to* H*. Symptom is defined as a given element of sigma algebra* R*.*

This definition means than symptom is a statement about history and there might be several histories leading to the same symptom. The symptoms can be categorized as follows:

Symptoms based on expert knowledge can be expressed in form of rules about data. Typically using some thresholds or intervals. These symptoms can be calculated quickly since they do not use either first-principle or statistical models. An example in the HVAC systems is whether a coil changes inlet and outlet temperatures. For these systems, it is possible to combine the expert knowledge with uncertainty using fuzzy logic [7].

Symptoms based on first principles assume some explicit law describing the system or its part in form of equations and inequalities. If they are not satisfied, the symptom is alerted. In contrast to the expert rules, these symptoms can work with some unknown parameters that have to be estimated from historical data. Comprehensive summary is provided in [10].

Univariate statistical symptoms focus on a single observed variable and decides about its abnormal behavior, typically based on unusual values of an observed signal (e.g. variance or skewness). Verron [12] provides advanced univariate signal analysis using Bayesian networks.

Multivariate symptoms discriminate whether two or more observed variables are in their usual relationship. There are two possible approaches to this task. First, a set of variables is considered together and a symptom occurs if this bundle exhibits abnormal behavior. Examples are principle component analysis [2] or auto-associative neural networks [5], often used for sensor fault detection.

Another approach uses statistical models with multiple inputs for one variable. The symptom then is defined as situation when the variable has significantly different value than expected, given other state variables. As an example, we can mention [14] where 11 performance indices were modeled as polynomials of some observable variables.

$$S_i = I(\hat{Y}_{j,\text{lower}}(H) \geq Y_j \geq \hat{Y}_{j,\text{upper}}(H))$$

The availability and applicability of symptoms may vary from system to system. Above mentioned summary provides the decision maker with wide range of available tools.

3.2 Observation Model and Bayesian Filtration

The connection between observed system output (or symptoms derived from it) and its real state is the so-called observation model (see Definition 1). With a known observation model of the system, we can at each time step update probabilities of each fault and faultless state. The tool to do so is called the Bayesian filtration:

Theorem 1 (Bayesian Filtration)**.** *Let* $(\mathbf{x}^{(0)}, f, g)$ *be a dynamic system,* $t_{\max} \in \mathbb{N}$ *time horizon,* $\mathcal{F} = \{\mathrm{F}_i \subset \mathbb{R}^{N_x} | i \in \{0, \ldots, n\}\}$ *set of fault states,* $\mathcal{S} = \{\mathrm{S}_j \subset \mathrm{U}_y | j \in \mathcal{J}\}$ *finite set of possible symptoms,* $t \in (0, t_{\max}]$*,* $H(t)$ *= history, where* $\mathbf{y}^{(t)} \in \mathrm{S}_k$*,* $k \in \mathcal{J}$*. Then*

$$\mathbf{Pr}(\mathbf{x}^{(t)} \in \mathrm{F}_i | H(t)) = \frac{\mathbf{Pr}(\mathbf{y}^{(t)} \in \mathrm{S}_k | \mathbf{x}^{(t)} \in \mathrm{F}_i) \mathbf{Pr}(\mathbf{x}^{(t)} \in \mathrm{F}_i | H(t) \setminus \mathbf{y}^{(t)})}{\int_{\mathrm{U}_x} \mathbf{Pr}(\mathbf{y}^{(t)} \in \mathrm{S}_k | \mathbf{x}^{(t)} \in \mathrm{F}_i) \mathbf{Pr}(\mathbf{x}^{(t)} \in \mathrm{F}_i | H(t) \setminus \mathbf{y}^{(t)}) \mathrm{d}\mathbf{x}} \tag{1}$$

Proof. [8] We will use the fact that

$$H(t) = \left(H(t) \setminus \mathbf{y}^{(t)}\right) \cup \mathbf{y}^{(t)}.$$

This implies

$$\mathbf{Pr}(\mathbf{x}^{(t)} \in \mathrm{F}_i | H(t)) = \mathbf{Pr}(\mathbf{x}^{(t)} \in \mathrm{F}_i | H(t) \setminus \mathbf{y}^{(t)}, \mathbf{y}^{(t)}),$$

and the theorem results from the Bayes rule. □

Bayesian filtration updates our belief on system state from belief based on history $H(\tau - 1)$ to belief based on history $H(\tau)$. The belief in time instant $\tau = 0$ is called the prior belief. In this version of Bayesian Filtration theorem, we first need to derive probabilities $\mathbf{Pr}(\mathbf{y} \in \mathrm{S}_i | \mathbf{x} \in \mathrm{F}_j)$ from the observation model $g(\mathbf{y}|\mathbf{x})$. This, however, is not a problem, because

$$\mathbf{Pr}(\mathbf{y} \in \mathrm{S}_i | \mathbf{x} \in \mathrm{F}_j) = \int_{\mathrm{S}_j} p(\mathbf{y}|\mathbf{x} \in \mathrm{F}_j) \mathrm{d}\mathbf{y},$$

$$p(\mathbf{y}|\mathbf{x} \in \mathrm{F}_j) = \int_{\mathrm{F}_j} g(\mathbf{y}|\mathbf{x}) \mathrm{d}\mathbf{x},$$

leading to

$$\mathbf{Pr}(\mathbf{y} \in \mathrm{S}_i | \mathbf{x} \in \mathrm{F}_j) = \int_{\mathrm{S}_i} \int_{\mathrm{F}_j} g(\mathbf{y}|\mathbf{x}) \mathrm{d}\mathbf{x} \mathrm{d}\mathbf{y}.$$

Thus the denominator in equation 1 can be written as a

$$\sum_{j=0}^{n} \mathbf{Pr}(\mathbf{y}^{(t)} \in \mathrm{S}_k | \mathbf{x}^{(t)} \in \mathrm{F}_j) \mathbf{Pr}(\mathbf{x}^{(t)} \in \mathrm{F}_j | H(t) \setminus \mathbf{y}^{(t)}).$$

The problem consists in obtaining the observation model itself – since we cannot directly observe the system states, we also do not have the precise probability density function $g(\mathbf{y}|\mathbf{x})$.

One way of obtaining the observation model is to model the whole system and do a simulation of its faultless run and then gradually introduce to the system all faults. For each fault F_i symptoms are observed and from the dataset the number of observed output in symptom S_j divided by the number of all observed outputs then represents the value of a conditional probability $\mathbf{Pr}(\mathbf{y} \in \mathrm{S}_j | \mathbf{x} \in \mathrm{F}_i)$, and therefore can be taken as a part of the system observation model used in Theorem 1. This approach is used for example in [14].

Another way, how to get an observation model for a given system, is to extract the knowledge from experts. It is also possible to combine different approaches, e.g. use expert knowledge for the prior estimate of the observation model and then refine it gradually by observed data, using Bayesian approach [8].

3.3 Dynamic Model and Prediction

The dynamic model of the system describes, how chosen inputs influence next system state. With a known dynamic model, we can predict system state in next time step, given current belief and selected input. This can be done via the so-called Bayesian prediction:

Theorem 2 (Bayesian Prediction)**.** *Let* $(\mathbf{x}^{(0)}, f, g)$ *be a dynamic system,* $t_{\max} \in \mathbb{N}$ *time horizon,* $\mathcal{F} = \{\mathrm{F}_i \subset | i \in \{0, \ldots, n\}\}$ *set of faults,* $t \in (0, t_{\max}]$*,* $H(t)$ *history, where* $\mathbf{y}^{(t)} \in \mathrm{S}_k$*,* $k \in \mathcal{J}$ *. Then*

$$\mathbf{Pr}(\mathbf{x}^{(t+1)} \in \mathrm{F}_i | H(t), \mathbf{u}^{(t)}) = \sum_{j=0}^{n} \mathbf{Pr}(\mathbf{x}^{(t+1)} \in \mathrm{F}_i | \mathbf{u}^{(t)}, \mathbf{x}^{(t)} \in \mathrm{F}_j) \mathbf{Pr}(\mathbf{x}^{(t)} \in \mathrm{F}_j | H(t)) \tag{2}$$

Proof. The theorem results directly from law of total probability. □

3.4 Dynamic Programming

With a dynamic system, given loss function and known dynamic and observation model, we can at each time step compute the best possible input in order to minimize the overall expected loss and so make an optimal maintenance strategy. The tool to do so is called the dynamic programming:

Theorem 3 (Dynamic Programming)**.** *Let* $(\mathbf{x}^{(0)}, f, g)$ *be a dynamic system,* $t_{\max} \in \mathbb{N}$ *time horizon,* z *partial loss function. Then the optimal strategy* R *can be computed this way: for each time instant* τ *and each belief* $B(\tau)$ *the input minimizing the Equation (3) is chosen:*

$$\nu(B(\tau), \tau) = \min_{\mathbf{u}^{(\tau)} \in U_{\mathbf{u}}} E[z(\mathbf{x}^{(\tau)}, \mathbf{u}^{(\tau)}) + \nu(B(\tau + 1), \tau + 1 | \mathbf{u}^{(\tau)})]. \tag{3}$$

Presented recursion is conducted from time horizon $\tau = t_{\max}$ *to* $\tau = 0$*, where*

$$\nu(B(t_{\max} + 1), t_{\max}) \equiv 0$$

for all beliefs $B(t_{\max} + 1)$*.*

Proof. Proof will be conducted by mathematical induction from time horizon $t = t_{\max}$ to $t = 0$. Let R_t denote strategy from time instant t to the time horizon. Let first $t = t_{\max}$ and R_t be the strategy constructed with the use of this theorem. Let R_t be not optimal strategy. This implies existence of strategy $\tilde{R}_t$ such that for one ore more beliefs $B(t)$ applies the inequality

$$\sum_{\mathbf{x}} z(\mathbf{x}, R(B(t), t_{\max})) \cdot \mathbf{Pr}(\mathbf{x}) > \sum_{\mathbf{x}} z(\mathbf{x}, \tilde{R}(B(t), t_{\max})) \cdot \mathbf{Pr}(\mathbf{x}),$$

where the probabilities $\mathbf{Pr}(\mathbf{x})$ are the subjective probabilities from belief $B(t)$. However, this is contradiction with the Equation 3, because clearly the minimizing input was not put into the strategy.

Let R_t be the optimal strategy from time t to $t_{\max}$ constructed with the use of this theorem. Let R_{t-1} be the extension to time instant $t-1$, also corresponding to this theorem, and let it be not optimal. Analogically it can be proven, that this results into contradiction with the Equation 3, so that such strategy must be optimal. □

The function ν from the Theorem 3 is called the Bellman function or the optimal loss-to-go and it has the meaning of optimal expected loss from given situation (time instant and belief) to the time horizon. With the use of Theorems 1, 2 and 3 we are now able to present Algorithm 1, a tool for calculating the optimal maintenance strategy for a given system.

The Algorithm 1 follows directly the Theorem 3: for each time step (computing backwards from time horizon) and belief minimal loss from that situation to time horizon is found and saved to the optimal strategy with corresponding input. The transitions from one situation to other are computed with the help of Bayesian filtration and prediction (Theorems 1 and 2).

A Bayesian belief $B(t)$ in general has infinite amount of values, which would lead to Algorithm 1 being not resultative. Therefore, a discretization of Belief is required, using a given discretization step (e.g. 0.01), so the probabilities in $B(t)$ can have only finite amount of values. The number of symptoms and faults is also finite (see Definitions 4,6) and we assume that the number of possible inputs (actions) is also finite. Therefore in each time instant we compute and compare finite amount of numbers and the algorithm is resultative.

The application of the discretization leads to moderate loss of information, but there are two justification for it: first, the measurements of observed variables in system (temperatures, velocities, etc.) and therefore the system outputs are always performed with finite accuracy; and second, too close system states have almost identical nature and are not distinguishable for the loss function in real applications.

Dynamic programming is an elegant way to find the optimal maintenance strategy with linear complexity with respect to the time horizon. The complexity of Algorithm 1 with respect to belief discretization and number of system faults $|\mathcal{F}|$ is far from linear, though: as shown in Figure 2, the complexity with respect to $|\mathcal{F}|$ is exponential ($O(\exp \mathcal{F})$) for any given discretization step (in Figure 2 for discretization step 0.1), because the number of possible beliefs $B(t)$ in any given time instant is (roughly) exponentially growing and the algorithm must examine every possible $B(t)$ for all time instants from time horizon to beginning.

Finally the complexity with respect to belief discretization is $O(p^{|\mathcal{F}|})$ for any given number of possible faults $|\mathcal{F}|$, where p is the number of points the interval $[0, 1]$ is divided into, so e.g. for discretization step 0.1 $p = 11$ as points are $0, 0.1, 0.2, \ldots, 0.9, 1$. This means that the complexity linked with computation accuracy is polynomial – however, for nontrivial systems with many possible faults the degree is uncomfortably high.

The problem of exponential complexity with respect to $|\mathcal{F}|$ and connected high degree polynomial complexity with respect to belief discretization is called the curse of dimensionality. The first two ways to fight it are:

- Keeping as few faults as possible – if some faults possible to happen in the system are e.g. of the same nature, have same consequences (and therefore costs) and are repaired by same or very close inputs, it might be helpful to agregate them, so that the number of possible faults $|\mathcal{F}|$ drops by 1,
- do the discretization as raw as possible – the discretization step should correspond to the sensor with smallest accuracy in order to prevent computing belief situations in the strategy, which cannot occur due to the output measurements inaccuracy.

However, the two methods presented above have serious limits as even the reduced number of faults can be high in more complex systems. In such situations it is appropriate to some of the methods presented in Section 4.

Algorithm 1 Optimal maintenance - strategy calculation

function CalculateStrategies
  for $\tau = t_{\max}, t_{\max} - 1, \ldots, 0$ **do**
    for all $B(t)$ **do**
      Loss $\leftarrow +\infty$
      for all u do
        if (Bellman($B(t)$, **u**) < Loss) **then**
          Loss $\leftarrow$ Bellman($f_{\mathbf{x}}$,**u**,t)
          Input $\leftarrow$ **u**
        end if
      end for
      Strategy $\leftarrow$ save [(t,$B(t)$)$\rightarrow$(Input,Loss)]
    end for
  end for
return Strategy
end function

function Bellman($B(t)$, **u**, t)
  Loss $\leftarrow Ez(\mathbf{x}, \mathbf{u}) = z(\mathbf{x}, \mathbf{u}) \cdot B(t)$
  $B'(t+1) \leftarrow$ Prediction($B(t)$,**u**)
  Loss $\leftarrow$ Loss + $\sum_{j=0}^{|S|} \mathbf{Pr}(\mathbf{y}^{(t)} \in S_j) \cdot$ Strategy$(t+1,$ Filtration$(B'(t+1), S_j))$.getLoss
return Loss
end function

function Filtration($B(t) = \{\mathbf{Pr}(\mathbf{x}^{(t)} \in \mathrm{F}_i | H(t)) | i = 0, \ldots, n\},\ S_j$)
  for $i = 0, \ldots, n$ **do**
    $\mathbf{Pr}(\mathbf{x}^{(t)} \in \mathrm{F}_i | H(t)) \leftarrow \frac{\mathbf{Pr}(\mathbf{y}^{(t)} \in S_j) \cdot \mathbf{Pr}(\mathbf{x}^{(t)} \in \mathrm{F}_i)}{\sum_{k=0}^{|S|} \mathbf{Pr}(\mathbf{y}^{(t)} \in S_j | \mathbf{x}^{(t)} \in \mathrm{F}_k) \cdot \mathbf{Pr}(\mathbf{x}^{(t)} \in \mathrm{F}_k)}$
    $B'(t) \leftarrow$ save $\mathbf{Pr}(\mathbf{x}^{(t)} \in \mathrm{F}_i | H(t))$
  end for
return $B'(t)$
end function

function Prediction($B(t)$, **u**)
  for $i = 0, \ldots, n$ **do**
    $\mathbf{Pr}(\mathbf{x}^{(t+1)} \in \mathrm{F}_i | H(t), \mathbf{u}^{(t)}) \leftarrow \sum_{j=0}^{n} \mathbf{Pr}(\mathbf{x}^{(t+1)} \in \mathrm{F}_i | \mathbf{u}^{(t)}, \mathbf{x}^{(t)} \in \mathrm{F}_j) \mathbf{Pr}(\mathbf{x}^{(t)} \in \mathrm{F}_j | H(t))$
    $B'(t+1) \leftarrow$ save $\mathbf{Pr}(\mathbf{x}^{(t+1)} \in \mathrm{F}_i | H(t), \mathbf{u}^{(t)})$
  end for
return $B'(t+1)$
end function

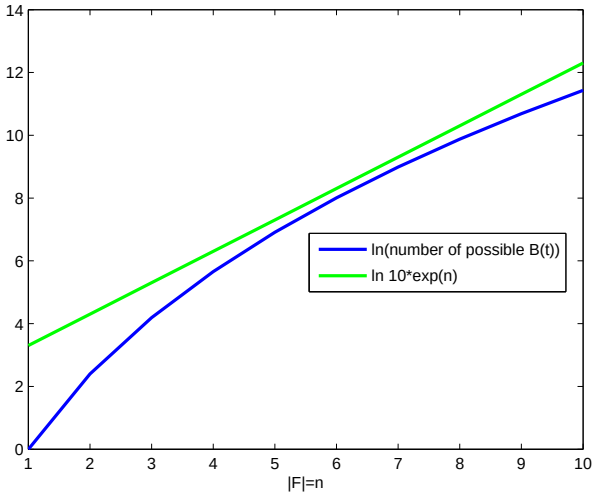

Figure 2: The number of all possible beliefs $B(t)$ with respect to the number of all possible faults $|\mathcal{F}|$ (logaritmic scale)

The main advantage of Algorithm 1 is that it has to be run only once, before the actual maintenance of the system. Once the optimal strategy is computed and saved, it is then only needed in each time step of the maintenance to compute new belief from sensors measurements (system outputs) and then find the optimal input in prepared strategy.

There is also the question of the shape of optimal strategy. As it turns out, the shapes of optimal maintenance strategies look very alike for different systems not only with different dynamic and observation models, but also with different dimensions (number of possible faults). From closer examination of computed optimal strategies could therefore evolve a different method for finding them based on setting up some parameters of common shape. Such method is outlined in Section 4.

With maintenance calculated, the simulation of a system run looks like presented in Algortihm 2.

Algorithm 2 Optimal maintenance - application of the strategy

```
loss = 0
belief ← initial belief
for τ = 0, 1, ..., t_max do
    belief ← Filtration(belief,output)
    u^(τ) ← Strategy(τ, belief)
    belief ← Prediction(belief, input)
    loss ← loss + z(x^(τ), u^(τ))
end for
```

4 Approximate Methods

The presented algorithm of dynamic programming offers optimal solutions to the system maintenance. However, for non-trivial systems the introduced curse of dimensionality is a serious issue and the basic algorithm begins to be computationally intractable. In this section we describe some less time consuming methods for finding close to optimal maintenance strategies.

4.1 Saving Filtration and Prediction Results

This approach tries to reduce the number of calling the functions of filtration and predictions in Algorithm 1, which is:

$$\text{number of calling filtration} = t_{\max} \cdot |\mathrm{U}_{\mathbf{u}}| \cdot |\mathcal{S}| \cdot \text{number of possible beliefs},$$

$$\text{number of calling prediction} = t_{\max} \cdot |\mathrm{U}_{\mathbf{u}}| \cdot \text{number of possible beliefs}.$$

These procedures can be quite time consuming. This method computes the filtration and prediction for each possible belief and each possible output and input in advance and saves the results into structure with constant time accessibility. The numbers of calling these functions drops to

$$\text{number of calling filtration} = |\mathcal{S}| \cdot \text{number of possible beliefs},$$

$$\text{number of calling prediction} = \mathrm{U}_{\mathbf{u}}| \cdot \text{number of possible beliefs}.$$

Only with this improvement, more than 99 % acceleration[2] of the dynamic programming algorithm were observed for some systems, resulting in same or very close strategies (different in less than 3 % of strategy inputs). The slight difference of values in strategies computed with the use of this method results from more rounding in this approach, as each resulting belief is rounded to a "possible" one (possible by discretization used).

4.2 Systems with Observable States

For systems with directly observable states, the Algorithm 1 is much simpler, as the strategy has to be computed for each combination of time instant and state, not time instant and belief. For example in system with four possible states, time horizon 1000 and belief discretization step 0,1 it is a difference between computing 286 000 and 4 000 values.

In a system with hidden states, the decision maker can still make use of the maintenance strategy computed for a system with the same dynamic model and loss function, but with observable states. Experiments were made with two approaches:

- The decision makers select input from the strategy corresponding to the state with highest probability in actual belief,
- they select input from the strategy corresponding to the state with highest value of Bellman function multiplied by the state probability in actual belief (mean of the Bellman function).

The first approach seems to bring better results, as the second one is too cautious of states with high possible losses. However, both approaches are good enough only for systems with strong diagnostics, that changes the actual belief quickly and with very few errors.

The observability of the states can be emulated by some smooth filtration like Kalman or particle filters [3]. Having the filtered estimate, faults can be evaluated by their definition straightforward.

4.3 Constant Border Strategies

All optimal strategies computed by Algorithm 1 for systems with functioning diagnostics, i.e. when precise observation and dynamics models are available, and inputs corresponding to faults (the system had the same number of inputs and states, each input repairing one state and one input having the meaning of no repair) had a common shape: constant border between inputs from beginning to a breakpoint close to time horizon, where the repairs became uneconomic and were deprecated. For a system without functioning diagnostics (the observation model was uniformly distributed) the optimal strategy had a shape of periodical repairs, as one would intuitively guess. See the Figures 3 and 4 for examples of optimal strategies for system with two states and two inputs. On x-axis is the time instant (with the time horizon of 100), on y-axis is the probability of being in state F_0 (faultless state). The blue area represents Belief values in time of the strategy suggesting repair of the system.

The common shape of optimal strategies implies another method for system maintenance: simply set the constant border between inputs and find the breakpoint t_z. Finding the breakpoint is very simple and results from the partial loss function. Under the condition, that such functions $z_{\mathbf{x}}$, $z_{\mathbf{u}}$ exist, so that

$$z(\mathbf{x}, \mathbf{u}) = z_{\mathbf{x}}(\mathbf{x}) + z_{\mathbf{u}}(\mathbf{u}),$$

the breakpoint t_z for state $\mathbf{x}$ and corresponding repair input $\mathbf{u}$ is the minimal natural number fulfilling the inequality

$$(t_{\max} - t_z + 1) \cdot z_{\mathbf{x}}(\mathbf{x}) > z_{\mathbf{u}}(\mathbf{u}),$$

E.g. for a state with corresponding loss 1 and its repair with the loss of 10, the last 10 time instants it is uneconomical to repair the system even if the decision maker is absolutely sure, that the system is in the faulty state.

The setup of the constant border from beginning to the computed breakpoint has not been fully explored yet. Observations of optimal strategies from Algorithm 1 suggest that it does not depend on

[2] Used case study for the results is presented in Section 6

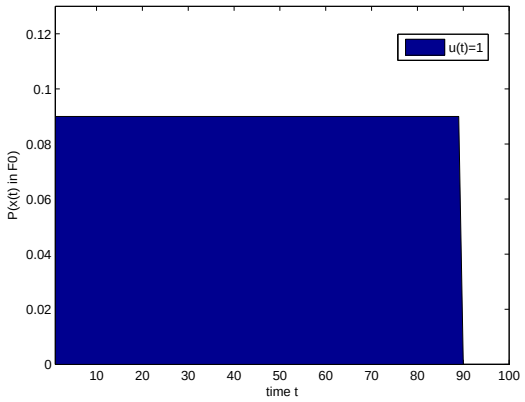

Figure 3: Maintenance Strategy for System with Functioning Diagnostics

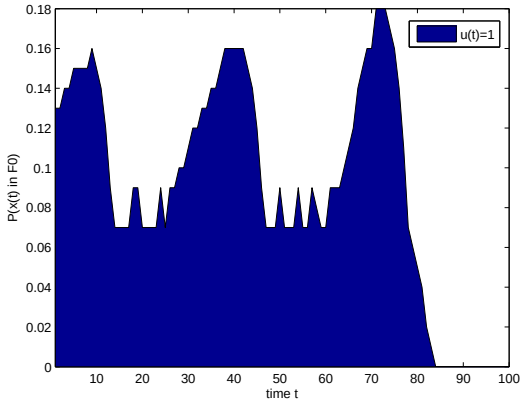

Figure 4: Maintenance Strategy for System without Diagnostics

time horizon, if it is high enough and finite. It depends on the loss function (the more expensive repair of a fault, the more must the decision maker be sure about the fault to repair the system) and dynamic and observation model of the system. The question of exact dependencies of these quantities remains open at this time.

However, simple policy for system with inputs corresponding to states was tested on few systems with pleasing results. The border for input $\mathbf{u}_i$ (repair of the fault F_i) was set as

$$1 - \frac{z_{\mathbf{x}}(\mathbf{x} \in \mathrm{F}_i)}{z_{\mathbf{u}}(\mathbf{u}_i)},$$

leading to the average loss only 0.72 % higher for a system with four states and time horizon of 100, than optimal strategy computed for the same system with the use of Algorithm 1 and discretization step of 0.05.

4.4 Approximate Dynamic Programming

All methods so far had an off-line character, as the maintenance strategies were set up before the run of the system. The method of approximate dynamic programming is an on-line approach: the maintenance strategy is computed on-line, constantly adjusted and revisited during it.

Approximate dynamic programming[3] is presented in Algorithm 3. Unlike the Algorithm 1, it does not find the strategy as a mapping of time instant and belief to inputs – instead its result is a fixed sequence of inputs. The application of the ADP in an on-line system maintenance run is presented in Algorithm 4. The $f_{\mathbf{x}}$ denotes the belief of a system state $B(t)$.

[3]For comprehensive information on ADP see [9].

Algorithm 3 Approximate Dynamic Programming

function ADP($\mathbf{x}_1^{(0)}$, $f_\mathbf{x}$, $\nu_0^{(0)}$,...,$\nu_0^{(t_\max)}$, f, $t_\max$, z, N)
for $n = 1, \ldots, N$ **do**
for $t = 0 : t_\max$ **do**

$$\begin{aligned}\mathbf{u}_n^{(t)} &\leftarrow \arg\min_\mathbf{u}[\sum_{\mathbf{x}'} z(\mathbf{x}', \mathbf{u}) \cdot f_\mathbf{x}(\mathbf{x}') + \\ &+ \sum_\mathbf{y} \nu_n^{(t+1)}(\text{fil}(\text{pred}(f_\mathbf{x}, \mathbf{u}), \mathbf{y})) \cdot \sum_{\mathbf{x}'} g(\mathbf{y}, \mathbf{x}') \cdot \text{pred}(f_\mathbf{x}, \mathbf{u})(\mathbf{x}'))]\end{aligned}$$

$$\begin{aligned}\nu &\leftarrow \min_\mathbf{u}[\sum_{\mathbf{x}'} z(\mathbf{x}', \mathbf{u}) \cdot f_\mathbf{x}(\mathbf{x}') + \\ &+ \sum_\mathbf{y} \nu_n^{(t+1)}(\text{fil}(\text{pred}(f_\mathbf{x}, \mathbf{u}), \mathbf{y})) \cdot \sum_{\mathbf{x}'} g(\mathbf{y}, \mathbf{x}') \cdot \text{pred}(f_\mathbf{x}, \mathbf{u})(\mathbf{x}'))]\end{aligned}$$

$$\nu_n^{(t)}(f_\mathbf{x}) = \begin{cases} \nu & f_\mathbf{x} = f_\mathbf{x}^{(t)} \\ \nu_{n-1}^{(t)} & f_\mathbf{x} \neq f_\mathbf{x}^{(t)} \end{cases}$$

$\mathbf{x}_n^{(t+1)} \leftarrow \text{random}(f(\cdot, \mathbf{u}_n^{(t)}, \mathbf{x}_n^{(t)}))$
$\mathbf{y}_n^{(t+1)} \leftarrow \text{random}(g(\cdot, \mathbf{x}_n^{(t+1)}))$
$f_\mathbf{x} \leftarrow \text{fil}(\text{pred}(f_\mathbf{x}, \mathbf{u}_n^{(t)}), \mathbf{y}_n^{(t+1)})$
end for
end for
return $\mathbf{u}_N^{(0)}, \mathbf{u}_N^{(1)}, \ldots, \mathbf{u}_N^{(t_\max)}$
end function

The ADP function starts from a given belief of the system and in each step finds an input minimizing the Bellman function, like the normal dynamic programming, but there are some differences: the Bellman function is not recursively and exactly computed. Instead, the ADP has the prior estimate of Bellman function values as its input. The ADP also proceeds in time from beginning to the time horizon. The main idea is a Monte Carlo simulation of the system run from the given point. At each time instant the algorithm makes a decision and updates the Bellman function. With more and more system run simulations, the estimate of the Bellman function is becoming more accurate and the ADP algorithm makes more optimal decisions (system inputs). A number of iterations can be decreased by good prior estimate. However, it is a non-trivial task as the estimate should not only be accurate, but also reasonably fast to make the difference.

The actual system run goes like this: At first, the input sequence is computed with the Algorithm 3 and prior belief. Then the system begins to run, inputs are sequentially chosen from the sequence and once in a while the sequence is computed again, with actual belief and lower time horizon. This can be interpreted as strategy revision. The ADP on-line maintenance unsurprisingly brings better results with more iterations N and more frequent strategy revisions (low interval k).

5 Reference Problem

With known system models and loss function, its maintenance is a question of optimal trade off between computational complexity and requirements on optimality of the strategy. In principle, however, it is easily manageable with tools presented in previous sections. The decision makers can choose to use the dynamic programming for finding optimal solutions. When the system is too complex and the curse of dimensionality becomes a serious issue, they have at least four other methods to deal with the maintenance task.

The main problem is therefore an accurate estimate of the observation model, the dynamic model, and the loss function. If the system has hidden states, it is clear that all the decision maker can hope for are estimates close to reality enough to bring satisfactory results. The first step in order to obtain system models is to find all possible states (and inputs and outputs) and derive them into faults, e.g. by fault tree diagnostics [4].

Algorithm 4 On-line System Maintenance Using ADP

function SIMULATEADP($t_{\max}, f, z, N, k, \mathbf{x}^{(0)}$)
  $\mathbf{u}^{(0)}, \mathbf{u}^{(1)}, \ldots, \mathbf{u}^{(t_{\max})} \leftarrow \mathrm{ADP}(\mathbf{x}^{(0)}, \nu_0^{(t)}, f, t_{\max}, z, N)$
  $Z \leftarrow z(\mathbf{x}^{(0)}, \mathbf{u}^{(0)})$
  for $t = 1, \ldots, t_{\max}$ **do**
    $\mathbf{x}^{(t)} \leftarrow \mathrm{random}(f(\cdot, \mathbf{u}^{(t-1)}), \mathbf{x}^{(t-1)})$
    if $t \bmod k = 0$ **then**
      $\mathbf{u}^{(t)}, \ldots, \mathbf{u}^{(t_{\max})} \leftarrow \mathrm{ADP}(\mathbf{x}^{(t)}, \nu_0^{(t)}, f, t_{\max} - t + 1, z, N)$
    end if
    $Z \leftarrow Z + z(\mathbf{x}^{(t)}, \mathbf{u}^{(t)})$
  end for
return Z
end function

With known system states, inputs and outputs, the next step is to find the models. This can be done by building physical model of the system, by statistical estimates from observed data in the original system or by using expert knowledge. These approaches can be combined, e.g. expert knowledge is used to bring an prior estimate or just some information about models, including types of considered distribution. The estimate is then specified with modeling of the physical processes in the system and further continually refined by observed data from its run.

Another issue is formulation of the loss function. For maintenance, its values have mostly the meaning of financial cost of state and system input, and the partial loss function can be therefore split into two parts:

$$z(\mathbf{x}, \mathbf{u}) = z_{\mathbf{x}}(\mathbf{x}) + z_{\mathbf{u}}(\mathbf{u}),$$

where $z_{\mathbf{x}}$ denotes the costs of states and $z_{\mathbf{u}}$ the costs of inputs. The part $z_{\mathbf{u}}$ can be formulated rather easily, as the cost of a repair consists of the cost of components needed, work of the servicemen and sometimes also the loss resulting from stopping the system for the repair. The part $z_{\mathbf{x}}$, however, is for some systems hardly computed. In HVAC systems for commercial buildings the cost of a fault state results mainly from discomfort of the building occupants: e.g. in an office, fault in HVAC system causes decrease of indoor air quality and the thermal control. This results into the staff working less efficiently, less office hours and probably getting ill more often. These changes are real, but very small, and to compute the resulting cost of the HVAC fault state needs a large quantity of detailed data. Faults of air conditioning systems in manufacturing are more simple to compute, as they can result into stopping the work processes and even throwing away a quantity of produced products (e.g. in pharmaceutical industry).

As for models of the system, the expert knowledge is usable for formulation of the loss function, too. Possible procedure is to compare the costs of inputs and states pairwise (similarly to setting the weights of criteria in the AHP method [11]) and define the loss functions $z_{\mathbf{x}}$, $z_{\mathbf{u}}$ from the ratios, as for the maintenance task the ratios between losses for states and inputs are important, not their absolute values.

6 Maintenance of an HVAC System

This section demonstrates tools and concepts described in previous sections on diagnostics and maintenance of a HVAC system. First, the system itself and construction of its models and loss function are introduced. Then optimal maintenance strategy is computed with the use of Algorithm 1 and its results are compared with other methods presented in this paper.

The discussed system is taken over from [14] and is a HVAC system of a commercial building in Kowloon, Hong Kong. It consists of five parts: cooling towers, chillers, secondary pumps before heat exchangers, heat exchangers, secondary pumps after heat exchangers. Diagram of the system is in Figure 5.

6.1 Models and Loss Function

In [14] the authors diagnose the system by taking these steps:

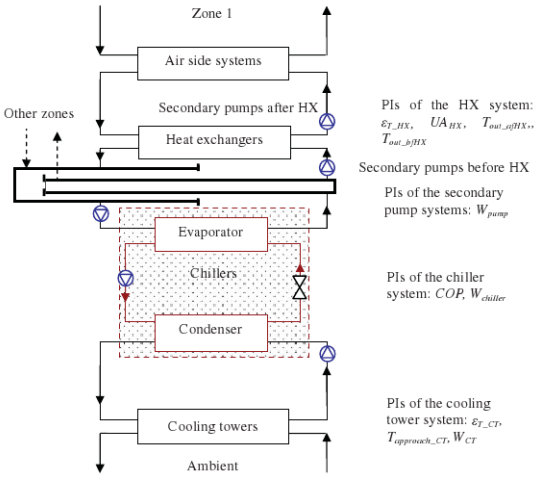

Figure 5: Diagram of Discussed HVAC System (taken from [14])

1. Model the system in the TRNSYS[4] (stands for Transient System Simulation Tool) environment
2. Define a number of performance indices (PI's) based on measurable values, e.g power consumptions or temperature effectiveness in the cooling system
3. By model simulation of healthy runs and runs with gradually introduced faults of two degrees of seriousness of the system they get a dataset of PI values under different situations
4. With this dataset they are able to predict PI values by quadratic regression
5. When the actual system is running, they measure the difference between predicted PI values and values computed from measured outputs. High difference suggests a fault in corresponding subsystem

Let us now formulate the system according to Definition 1. For the sake of simplicity, only one subsystems, chillers (more precisely its heat transfer), will be taken into account. However, presented methods are general and can be used for the whole system. First, space of possible states, inputs and outputs must be determined. Heat transfer in chiller system can be in a faultless state, or can suffer from condenser and evaporator fouling. Two faults are recognized: a mild and serious fault. Therefore, the set $\mathcal{F}$ consists of three elements, F_0 denoting the faultless state, F_1 mild and F_2 the serious condenser and evaporator fouling.

Outputs are discrete values describing whether a PI residual is high enough to warn about a fault. The two PI's corresponding to heat transfer in chiller system are coefficient of performance and power consumption, so system output can be defined as a discrete vector with dimension of two.

In [14] only diagnostics is discussed. What to do with the information about faults present in the system is completely up to the decision maker. Therefore, the dynamic model and loss function formulation do not emerge from any dataset. However, they will be built with an inner logic instead of being just a set of random numbers. We define system input as a discrete vector with dimension of one, simply denoting whether the decision maker decides to repair the chiller system or not.

Detection ratios in Table 1 (also from [14]) were computed as number of warnings due to high residual between predicted and measured values of according PI divided by the number of all measurements under different system states (Lvl 1 signifies mild fault of a subsystem, while Lvl 2 denotes a more serious fault). This means that the values in Table 1 are conditional probabilities (in %) of outputs under system state – an observation model from Definition 1. The only numbers missing are probabilities of outputs under the faultless state, therefore they are sadly to be defined artificially.

The loss function Z maps system states and inputs to a non-negative real value. As it has the meaning of financial cost, we can define only partial loss function z and even divide it into two functions $z_{\mathbf{x}}$, $z_{\mathbf{u}}$ for cost of states only and inputs only. As the input can have the values of "repair" and "do not repair". Under natural conditions doing nothing is free of cost; the cost of repair will be set to 10, so (denoting the repair as 1 and not repairing the system as 0):

$$z_{\mathbf{u}}(\mathbf{u} = 0) = 0,$$

[4] http://www.trnsys.com/

	Detection ratios (%)											
	Cooling tower system		Chiller system (motor)		Chiller system (heat transfer)		Secondary pump system before HX		Heat exchanger sytem		Secondary pump system after HX	
PI residual	Lvl 1	Lvl 2	Lvl 1	Lvl 2	Lvl 1	Lvl 2	Lvl 1	Lvl 2	Lvl 1	Lvl 2	Lvl 1	Lvl 2
$\epsilon_{\text{T\_CT}}$	30	100	8	20,67	12,67	20,67	18,67	18,79	12,67	12,67	18,67	12,67
$T_{\text{approach\_CT}}$	32	100	8	18,67	11,33	17,33	16	16,11	10	9,33	14	8,67
W_{CT}	100	100	14	18	12,67	17,33	17,33	16,78	20,67	20,67	16,67	16,67
COP	0	40	16,67	100	**36**	**70,67**	0	0	0	0	0	0
W_{chiller}	61,33	84	78,67	100	**80,67**	**89,33**	12	12,08	30	31,33	19,33	32
η_{HX}	0	0	0	0	0	0	0	0	0	100	0	0
UA_{HX}	4	5,33	4	2,67	0,67	3,33	0,67	3,36	100	100	9,33	23,33
$T_{\text{out\_afHX}}$	0	0	0	0	0	0	0	0	0	100	0	0
$T_{\text{out\_bfHX}}$	0	0	0	0	0	0	0	0	0,67	100	0	8,11
$W_{\text{pump\_bfHX}}$	0	8	8,67	7,33	8,67	8,67	100	100	2,67	0	17,33	18
$W_{\text{pump\_afHX}}$	0	0	0	0	0	0	0	0	0	0	35,33	99,33

Table 1: Detection Ratios for PI's for Faults in Different Subsystems

$$z_{\mathbf{u}}(\mathbf{u} = 1) = 10.$$

The system state can be in three sets F_0 for faultless state, F_1 for mild fault and F_2 for a serious one. A natural condition for this setting is

$$z_{\mathbf{x}}(\mathbf{x} \in \text{F}_0) \leq z_{\mathbf{x}}(\mathbf{x} \in \text{F}_1) \leq z_{\mathbf{x}}(\mathbf{x} \in \text{F}_2).$$

Let us for example define these costs as 0, 1 and 3. It is worth noting that absolute values are not essential for the loss function, as loss functions z and $c \cdot z$ lead to the same strategies for all $c > 0$. In other words, the ratios between values of loss function are important, not the absolute values.

The only missing piece for solving the maintenance task is the dynamic model of the system. We formulated it under three conditions:

- The state of the system cannot improve without repair
- The probability of mild fault is higher than probability of a serious one
- The system can break down right after repaired

We set the probability of inception a mild fault as 0.02, the probability of a serious one as 0.01 and the probability of changing the fault from mild to serious as 0.2. Along with the conditions mentioned above it is sufficient information for definition of the whole dynamic model.

6.2 Results of Different Maintenance Strategies

All tools presented in this paper were implemented in the Matlab environment and are available on `http://kmlinux.fjfi.cvut.cz/~berkaja3/dynsys/`. Let us first show the definition of system models and loss function discussed in previous subsection.

The output is a binary vector with dimension of two, i.e. there are four alternatives of its values (four possible symptoms). System state is a ternary (faultless state, mild fault, serious fault) vector with dimension of one. The observation model can be therefore written as a 4×3 matrix O, where

$$(O)_{ij} = \mathbf{Pr}(\mathbf{y} \in \text{S}_i | \mathbf{x} \in \text{F}_j).$$

Let us artificially define the probabilities of sensors pointing at a system fault under faultless state as 0.3 for both PI's. Under the condition of independent PI's (which seems reasonable as each PI is computed from different measurements) the observation model `O` can be in Matlab defined as:

```
O = [0.49 0.1216 0.0319;0.21 0.5184 0.2581;...
0.21 0.0684 0.0781;0.09 0.2916 0.6319];
```

As in each time step the output must have some realization, one can easily check the validity of the observation model by `sum(O)`. If `O` is a valid representation of an observation model, the result must be a vector of ones only.

Similarly, dynamic model for this system can be written as a $3 \times 2 \times 3$ matrix, as system state is ternary and input is binary vector with dimension of one. From the discussion in above emerges this definition in Matlab:

```
D = [0.97 0.02 0.01 0.97 0.02 0.01 0 0.8...
 0.2 0.97 0.02 0.01 0 0 1 0.97 0.02 0.01];
```

`D = reshape(D,3,2,3)`

The loss function will be 3×2 matrix defined as follows:

`L = [0 5; 1 6; 2 7];`

We will find strategies for maintenance of this system for 100 time instants, therefore we define time horizon as

`tmax = 100;`[5]

The last thing to do before running the algorithms described in Sections 3.4 and 4 is to do the discretization of probability. The belief for this particular system is a vector with length of three (values are belief of system being in a faultless state, of having a mild fault and of having a serious fault[6]). A function to get all the distinguishable beliefs was written also it Matlab, its input being the dimension of the system (number of all possible faults and faultless state) and the step (should be number which divides 1). The smaller step the more beliefs are distinguished and the more precise is the computation. For this example we set the step as 0.1 and get matrix of all beliefs by

`bels = getBeliefs(3, 0.1);`

To find the optimal strategy using the Algorithm 1 we run the `findStrategy` function:

`[str strLoss]= findStrategy(tmax, 0, D, L, bels);`

In strLoss are saved the values of the Bellman function, i.e. the optimal losses-to-go from given time step and belief.

The elapsed time was about 49 seconds (in Linux Ubuntu 9.10 on a machine with Intel Core2 T72002.00 GHz processor with 1 GB RAM. All computations were done at this machine unless stated otherwise). If we try to use the method described in Subsection 4.1, we must compute results of filtration and prediction in advance and then use them in the main algorithm:

```
OM = filtrationMat(0, bels);
DM = predictionMat(D, bels);
strM = findStrategyMat(tmax, 0, OM, DM, L, bels);
```

The elapsed time of `findStrategyMat` was 0.3 seconds, the two preparatory functions took about 0.5 seconds together. That means finding the maintenance strategy in less than 1 second for the same system. The `str` is an optimal strategy (with respect to the probability discretization). If we look closely on the `strM`, we find out, that it differs in 467 situations[7], i.e. in about 7 %. is exactly the same. For discretization step 0.01 the strategies differ in 0.2 % – the more precise computation, the less are the rounding in `findStrategyMat` "hurt" the optimality of strategy.

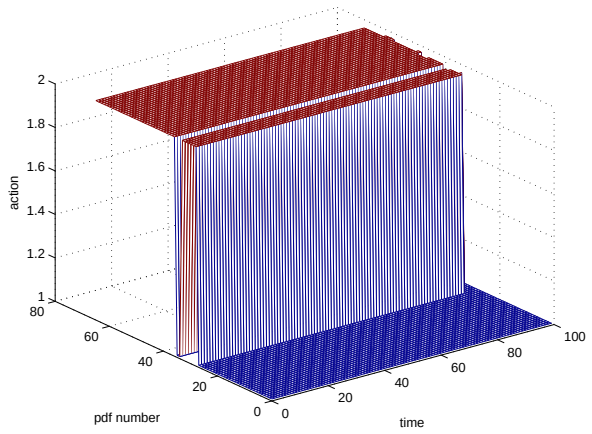

Figure 6: Optimal Maintenance Strategy for HVAC System

The average loss for a given system and strategy can be computed by simulations of system runs and averaging the achieved overall losses. As the expected loss is computed step-by-step in the dynamic programming algorithm, we can avoid this time consuming process and just look at the expected loss for the first time step. If the actual system models are equal to the models used in the Algorithm 1, the average loss from simulation converges to this value.

[5] In Matlab indexes run from 1, not from 0 as in theory introduced in Section 2.

[6] It is obvious that the sum of belief in each time step must be 1.

[7] This can be easily computed in Matlab e.g. by typing `sum(sum(str =strM))`

For strategies prepared by approximate algorithms, the expected losses are also approximate values and cannot be taken as a fact. However, for a given strategy the Bellman function can be computed ex-post. This gives us the opportunity to compare off-line strategies directly through comparison of values of the Bellman function in the first time instant (the belief that the system is certainly in faultless state will be used). Strategy `strM` achieves expected loss of 27.8775, while `str` 27.5982. The method of saving filtration and prediction results in advance produces slightly suboptimal maintenance strategy, but also saves a lot of computing time.

The shape of optimal maintenance strategy `str` is in Figure 6 (action 1 means doing nothing and action 2 repairing the system). It is just another example of shape discussed in Subsection 4.3. In this particular case the cutoff time step, after which it is uneconomic to repair system under any circumstances, is 3 steps before time horizon – as the repair costs 5 and serious fault 2, two or one step to go the cost of the repair would never be less than letting the system be in a fault state.

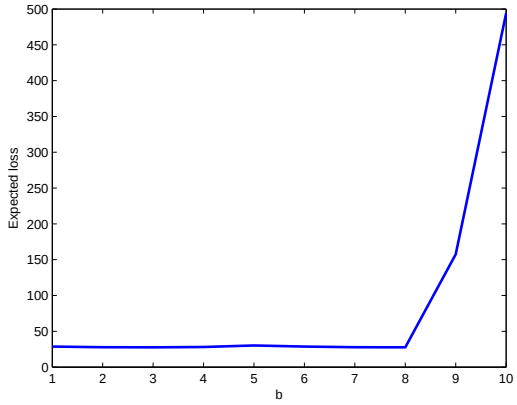

Figure 7: Expected Losses for Constant Border Strategies

According to method of setting constant borders, we define some strategies repairing the system, if the belief of the faultless state drops below a constant b and find the expected loss by computing the Bellman function for such strategy. The resulting losses are showed in Figure 7. The minimal reached value was 27.6795, unsurprisingly slightly higher than the loss for optimal strategy. The expected loss for strategy of not doing any actions ever was 125.106.

A naive approach completely omitting the diagnostics of the system would be a regular, periodic maintenance. The periods of 5, 10, ...50 were tested – the results are showed in Figure 8, reaching the minimum of 60.2858, more than two times of the optimal expected loss.

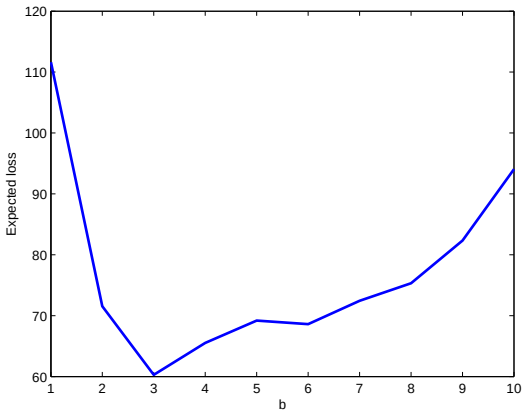

Figure 8: Expected Losses for Periodic Repair Strategies

The results presented in this section suggest the importance of maintenance strategy design: all strategies were computed for the same diagnostic model. Their losses, however, ranged from the minimum of about 28 through 60 for the best possible regular maintenance strategy to 125 for no

repairs. Even the constant border strategies can bring unsatisfactory results for bad choice of borders.

7 Conclusion

The paper has demonstrated capabilities of dynamic programming in the optimal maintenance, as well as other methods designed to overcome the curse of dimensionality. There remain, however, some open issues. The most serious one is the loss function which is hard to be defined properly in case of HVAC systems whose purpose is comfort. Comfort as such is relatively vague and difficult to be measured and even more difficult is to assign losses to the discomfort.

Another issue is the estimation of observation and dynamic model of the system. The more accurate are the models, the better results bring the strategies derived from them. It is hard to estimate the models accurately for dynamic systems with hidden states, as it takes a lot of insight into physical processes present in it and a large amount of data. One of possible improvements is to exploit the expert knowledge, but this approach has its own downsides, as not for all systems is the knowledge available and when it is, the transition of vague human knowledge into estimates of conditional probabilities is far from trivial.

Even though these aspects are worth to be assessed more properly, the article has summarized wide range of diagnostic methods and proposed several methods for optimized reasoning about the Bayesian belief. The variants provide the potential designers with various tools that can be adjusted for different systems and diagnostic needs.

References

[1] Francois Besnard. On stochastic dynamic programming and its application to maintenance. Master's thesis, KTH Electrical Engineering, 2007.

[2] Youming Chen and Lili Lan. Fault detection, diagnosis and data recovery for a real building heating/cooling billing system. *Energy Conversion and Management*, January 2010.

[3] Zhe Chen. Bayesian filtering: From kalman filters to particle filters, and beyond. Technical report, McMaster University, 2003.

[4] Thomas W. DeLong. *A fault tree manual*. National Technical Information Service, 1970.

[5] Huang Guo-Jian, Liu Gui-xiong, Chen Geng-xin, and Chen Tie-qun. Self-recovery method based on auto-associative neural network for intelligent sensors. In *World Congress on Intelligent Control and Automation (WCICA), 2010 8th*, 2010.

[6] Andrew K.S. Jardine, Daming Lin, and Dragan Banjevic. A review on machinery diagnostics and prognostics implementing condition-based maintenance. *Mechanical Systems and Signal Processing*, 20:1483–1510, 2006.

[7] Karel Macek, Jaromir Kukal, Jana Trojanova, and Jiri Rojicek. From symptoms to faults: Temporal reasoning methods. In *International Conference on Adaptive and Intelligent Systems*, 2009.

[8] V. Peterka. Bayesian approach to system identification. In *in Trends and Progress in System Identification, P. Eykhoff, Ed*, pages 239–304. Pergamon Press, 1981.

[9] W.B. Powell. *Approximate dynamic programming: solving the curses of dimensionality*. Wiley series in probability and statistics. Wiley-Interscience, 2007.

[10] Raymond Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32(1):57 – 95, 1987.

[11] Thomas L. Saaty. *The analytic hierarchy process: Planning, priority setting, resource allocation*. McGraw-Hill, New York and London, 1980.

[12] Sylvain Verron, Teodor Tiplica, and Abdessamad Kobi. Fault detection of univariate non-Gaussian data with Bayesian network. In *IEEE International Conference on Industrial Technology (ICIT'10)*, Vina del Mar, Chile, 2010.

[13] Larry Wasserman. *All of Nonparametric Statistics*. Springer Berlin / Heidelberg, 2006.

[14] Qiang Zhou, Shengwei Wang, and Zhenjun Ma. A model-based fault detection and diagnosis strategy for hvac systems. *International Journal of Energy Research*, 33:903–918, 2010.

Jan Berka
e-mail: berkaja3@fjfi.cvut.cz