Journal of Artificial Intelligence Research 65 (2019) 487-518 Submitted 02/19; published 08/19

Unifying System Health Management and Automated Decision Making

Edward Balaban EDWARD.BALABAN@NASA.GOV
NASA Ames Research Center
Intelligent Systems Division
Moffett Field, CA 94035 USA

Stephen B. Johnson STEPHEN.B.JOHNSON@NASA.GOV
Dependable System Technologies, LLC,
Westminster, CO 80234, USA

Mykel J. Kochenderfer MYKEL@STANFORD.EDU
Department of Aeronautics and Astronautics
Stanford University
Stanford, CA 94305 USA

Abstract

Health management of complex dynamic systems has evolved from simple automated alarms into a subfield of artificial intelligence with techniques for analyzing off-nominal conditions and generating responses. This evolution took place largely apart from the development of automated system control, planning, and scheduling (generally referred to in this work as decision making). While there have been efforts to establish an information exchange between system health management and decision making, successful practical implementations of integrated architectures remain limited. This article proposes that rather than being treated as connected yet distinct entities, system health management and decision making should be unified in their formulations. Enabled by advances in modeling and algorithms, we believe that a unified approach will increase systems' resilience to faults and improve their effectiveness. We overview the prevalent system health management methodology, illustrate its limitations through numerical examples, and describe a proposed unified approach. We then show how typical system health management concepts are accommodated in the proposed approach without loss of functionality or generality. A computational complexity analysis of the unified approach is also provided.

1. Introduction

The field of *system health management (SHM)* progressed from simple red-line alarms and human-initiated responses to a discipline that often includes sophisticated reasoning algorithms and automated fault recovery recommendations (Aaseng, 2001). The main goal of modern SHM has been defined as the preservation of a system's ability to function as intended (Johnson & Day, 2011; Rasmussen, 2008). While in this paper we apply the term SHM to the operational phase of a system's lifespan, the term may also encompass design-time considerations (Johnson & Day). The actual achievement of a system's operational objectives, on the other hand, is under the purview of the fields of control, planning, and scheduling. In this paper we will generally refer to all the processes aimed at accomplishing

operational objectives as *decision making (DM)*, while using more specialized terms where necessary.

Historically, the field of SHM has developed separately from DM. Typical SHM functions (monitoring, fault detection, diagnosis, mitigation, and recovery) were originally handled by human operators, similar to DM (Aaseng, 2001; Ogata, 2010). As simple automated control was being introduced for DM, automated fault monitors and alarms reduced operator workload for SHM. Gradually, more sophisticated control techniques were developed for DM (Ogata), while automated emergency responses started handling some off-nominal system health conditions (Rasmussen, 2008). Capable computerized planning and scheduling tools eventually became available for performing higher-level, more strategic decision making (Ghallab, Nau, & Traverso, 2016). On the SHM side, advanced methods were developed to improve robustness of anomalous event detection and alerting (Chryssanthacopoulos & Kochenderfer, 2012; Iverson, 2004). Computerized fault diagnosis was also added, eventually growing into a subfield of artificial intelligence in its own right (Feldman, Provan, & Gemund, 2010; Korbicz, Koscielny, Kowalczuk, & Cholewa, 2012; Metodi, Stern, Kalech, & Codish, 2014). In some cases diagnosis was coupled with failure prediction (*prognostic*) algorithms (Lu & Saeks, 1979), as well as with recovery procedures (Avizienis, 1976). Still, the two sides, SHM and DM, remained largely separated.

When the concept of *integrated system health management (ISHM)* became formalized, most interpretations of *integrated* encompassed some interactions between SHM and DM, although practical implementations of such interactions have been limited (Figueroa & Walker, 2018). A notable early example of ISHM is the Deep Space 1 Remote Agent Experiment, or RAX (Bernard et al., 1999). RAX integrated an automated planner/scheduler, a command execution system, and a mode identification and reconfiguration module. The latter performed state estimation and fault detection/diagnosis. If a fault was detected and identified, a single recovery action was recommended to the execution module. The mode identification and reconfiguration module was based on Livingstone (Williams & Pandurang, 1996), a reactive configuration management engine that reasons over declarative models. Livingstone later evolved into Livingstone 2 (Kurien & Nayak, 2000), which served as the diagnostic component of several other ISHM prototypes (Meyer et al., 2003; Nicewarner & Dorais, 2006; Schwabacher, Samuels, & Brownston, 2002). In Livingstone 2 problems were represented as partially observable Markov decision processes (Kaelbling, Littman, & Cassandra, 1998), although that representation was only used for tracking the state of a system, rather than for generating action recommendations. Benedettini, Baines, Lightfoot, and Greenough (2009) and Prajapati, Roy, and Prasad (2018) list additional ISHM implementation examples, while also noting that they have been relatively rare.

This paper makes the following claims:

1. For systems with controllable degradation processes, performing prognostics is not meaningful; for systems with uncontrolled degradation processes, prognostics may only be meaningful under certain conditions.
2. SHM should be unified with DM for greater operational effectiveness and resilience to system faults and there is a path for doing so.
3. Automated emergency response should be done separately from unified SHM/DM to provide performance guarantees and dissimilar redundancy.

For the second claim, we intend to show a path towards SHM/DM unification that builds on the latest developments in state-space modeling, planning, and control. In the past, limitations in computing hardware and algorithms would have made unified SHM/DM challenging to implement, but we believe that recent advances make that attainable. We also believe that this unified approach is applicable to a broad spectrum of DM, from traditional deterministic planners to complex algorithms computing long-horizon action policies in the presence of uncertainty. We use the term *unification* to emphasize the idea of SHM and DM being performed within the same framework, rather than the two being integrated as separate subsystems exchanging information.

Some initial progress towards SHM/DM unification can be found in prior work by Balaban and Alonso (2013), Balaban, Arnon, Shirley, Brisson, and Gao (2018), and Johnson and Day (2010, 2011). System health information was also incorporated into DM by others (Agha-mohammadi, Ure, How, & Vian, 2014; Spaan, Gonçalves, & Sequeira, 2010; Ure, Chowdhary, How, Vavrina, & Vian, 2013). This article aims to introduce a systematic view on such unification, discuss its benefits, and illustrate how current SHM concepts map into the proposed approach without loss of functionality or generality.

We first define the categories of systems that are of interest to this study (Section 2) and then give an overview of the prevailing approach to SHM (Section 3). The first claim (on prognostics) is discussed in Section 4. Section 5 discusses the second claim, concerning SHM/DM integration and its benefits, as well as the rationale for the third claim, that SHM/DM should be separate from automated emergency response. A computational complexity analysis of the prevailing methodology versus the proposed unified approach is presented in Section 6. Section 7 concludes.

2. Systems of Interest

We focus our attention on systems that have degradation processes affecting the system's performance within its expected useful life span. We consider both systems with uncontrolled degradation processes and those with controllable degradation processes. For our purposes, the *uncontrolled degradation* category includes not only those system types for which control over their degradation processes is not available or required, but also those operating on predefined control sequences, such as industrial robots performing the same sets of operations over extended periods of time. Also included are system types where degradation is considered uncontrolled within some time interval of interest (*decision horizon*). The rate of degradation is influenced by internal (e.g., chemical decomposition) and external factors (e.g., temperature of the operating environment). In addition to industrial robots, examples of system types with uncontrolled degradation processes include bridges, buildings, electronic components, and certain types of rotating machinery, such as electrical power generators. In systems within the *controllable degradation* category, degradation processes are influenced not only by the same types of internal and external factors as for the first category, but also by control actions, either directly or indirectly.

Most of our discussion is applicable to both categories, although systems with controllable degradation processes would, naturally, benefit more from active SHM/DM. In describing the systems, we adopt the notation from the field of decision making under uncertainty (Kochenderfer, 2015).

A system *state* s can be a scalar or a vector belonging to a state space S. For uncontrolled processes, a *transition model* $T(s, s')$ describes the probability of transitioning to a particular state $s' \in S$ from state s. For controllable processes, an *action* a initiates state transitions, with A denoting the space of all available actions (A may be state-dependent). A *transition model* for controllable processes takes the form $T(s, a, s')$, describing the probability of transitioning to a particular state $s' \in S$ as a result of taking action a from state s. A *reward model* $R(s, a)$ for controllable processes describes a reward obtained as a result of taking action a from state s. *Terminal states* form a subset $S_\text{T} \subset S$. Terminal states S_T may include both failure and goal states. Transitions from a terminal state are only allowed back to itself.

If there is state uncertainty (partial observability), *belief states* are used instead of regular states (also referred to as *beliefs*). A belief b is a probability distribution over S, with B denoting the space of all beliefs. *Observations* (e.g., sensor readings) can help with *belief estimation* and *belief updating*. Like a state, an observation can be a vector quantity. For uncontrolled processes, an *observation model* $O(s', o)$ describes the probability of an observation o being generated upon transition to state s'. For controllable processes, an *observation model* $O(s', a, o)$ does the same, but for a transition resulting from action a. In a partially observable setting, a *history* is either a sequence of observations $h_t = \{o_1, \ldots, o_t\}$ for uncontrolled processes or a sequence of actions and observations $h_t = \{a_1, o_1, \ldots, a_t, o_t\}$ for controllable processes.

The general function of decision making is to select actions. While in some systems we are only concerned with selecting a single a_t at a given time t, decision-making problems often involve selecting a sequence of actions. In a strictly deterministic system operating in a deterministic environment, an entire sequence of actions (a *plan*) can be selected ahead of time. In systems with action outcome uncertainty, however, a fixed plan can quickly become obsolete. Instead, a *policy* $\pi(s) : S \to A$ needs to be selected that prescribes which action should be taken in any state. In a partially observable setting, a policy maps beliefs to actions, i.e., $\pi(b) : B \to A$. A policy can be either *offline* (precomputed for all states or beliefs of interest) or *online* (computed for the current state or belief only). An optimal policy π^* is a policy that, in expectation, optimizes a desired metric (e.g., maximizes cumulative reward).

Throughout the paper, we use a robotic exploration rover operating on the surface of the Moon as the main running example of a complex system with controllable degradation processes. The rover is solar-powered and stores electrical energy in a rechargeable battery. Where it benefits the discussion, we also introduce other examples of systems and system components from the aerospace domain.

3. System Health Management

A typical contemporary SHM integration approach is shown in Figure 1. A DM subsystem generates an action $a_{t,\text{DM}}$, aimed at achieving operational objectives. The plant executes $a_{t,\text{DM}}$ and an observation o_t is generated and relayed to both DM and SHM. DM computes $a_{t+1,\text{DM}}$ on the basis of o_t, while SHM analyzes o_t for indications of *faults* (defined here as system states considered to be off-nominal) and, if any are detected, issues a recommendation on mitigation or recovery $a_{t+1,\text{SHM}}$ to the DM subsystem or, in some cases, as a

command directly to the plant (Valasek, 2012). SHM may select $a_{t+1,\mathrm{SHM}}$ from the general system action space A_{DM} or, if defined, from an additional A_{SHM} space containing only actions specific to system health.

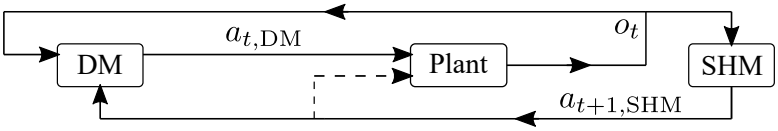

Figure 1: A typical system architecture with SHM

Figure 2 shows the SHM subsystem. The fault detection module corresponds to traditional red-line monitors detecting threshold-crossing events of sensor values, represented on the diagram by a Boolean *fault detection* function F. If a fault is detected ($F(o_t) = \mathrm{true}$), *fault isolation* and *diagnosis* (or *identification*) are performed, generating a vector of fault descriptors f_t (Daigle & Roychoudhury, 2010). Each fault descriptor typically identifies a component, its fault mode, and its fault parameters (Daigle & Roychoudhury). There are diagnostic systems that also include an estimated fault probability in the descriptor (Narasimhan & Brownston, 2007). If the uncertainty of the diagnostic results is deemed too high (e.g., f_t consists of only low-probability elements), *uncertainty management* is sometimes performed in order to obtain a better estimate of the current system condition (Lopez & Sarigul-Klijn, 2010).

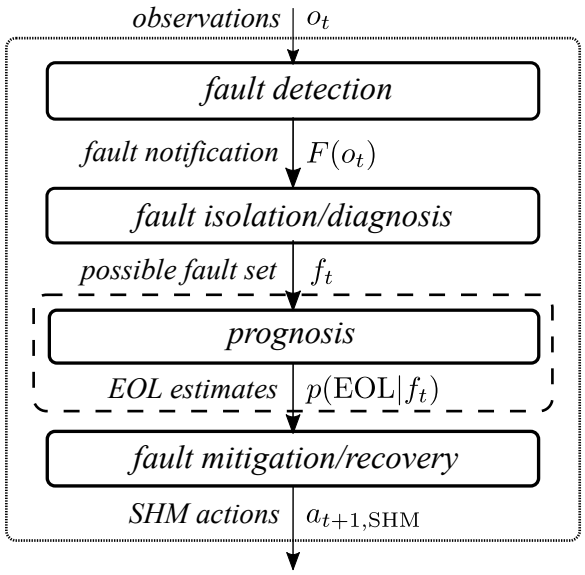

Figure 2: A typical contemporary SHM architecture

Some recent SHM implementations then pass f_t to a *prognostic* module (Roychoudhury & Daigle, 2011). In the SHM context, the intended goal of the prognostic module is to predict, at time t_{p} (here $t_{\mathrm{p}} = t$), whether and when faults will lead to system *failure* (defined as inability to perform its assigned function) within the window $[t_{\mathrm{p}}, t_{\mathrm{p}} + H]$ of a prediction horizon H (the terms *prediction horizon* and *decision horizon* are equivalent

for our purposes). In prognostics literature, time of failure is commonly synonymous with the term *end of [useful] life (EOL)*. Equivalently, the goal of prognostics can be defined as predicting the system's *remaining useful life (RUL)*. In Figure 2, the prognostic prediction is defined as a probability density of EOL given the set of system faults at time t, $p(\text{EOL}|f_t)$. Uncertainty management is sometimes also prescribed following prognostic analysis, meant to improve the prediction if the confidence in it is low (Wang, Youn, & Hu, 2012). If a prognostic module is part of an SHM sequence, the term *prognostics and health management (PHM)* is used by some instead of SHM in order to emphasize the role prognostics is playing in managing the system's lifecycle (e.g., Ferrell, 2000).

Finally, $p(\text{EOL}|f_t)$ and f_t are passed to the *fault mitigation and recovery* component to select an action $a_{t+1,\text{SHM}}$ from the action set A_{SHM} in order to mitigate or recover from faults in f_t. As part of this process, operational constraints may be set for those faulty components that cannot be restored to nominal health. If functional redundancy exists for such components, their further use may be avoided.

The overall limitations of the current SHM methodology are discussed in Section 5, where an approach that unifies SHM and DM is then proposed. The next section, however, focuses on the prognostic component and discusses why it is not meaningful for systems with controllable degradation processes and is challenging to implement in a useful manner for systems where degradation is uncontrolled.

4. Prognostics

A general definition of prognostics is that of a procedure that can predict the time of occurrence of an event E (Daigle, Sankararaman, & Kulkarni, 2015). Using notation from Section 2, if $\phi_E : S \to \mathbb{B}$ (where $\mathbb{B} \triangleq \{0, 1\}$) is an event threshold function, then $t_E \triangleq \inf\{t \in [t_\text{p}, t_\text{p} + H] : \phi_E(s_t) = 1\}$ is the nearest predicted time of E (s_t is the state at time t). If the state evolution trajectory is non-deterministic, then $p(t_E|s_{0:t_\text{p}})$ is computed instead. Figure 3 illustrates prognostics on an example of a system's EOL prediction, where the $p(\text{EOL})$ probability distribution is estimated based on a set of stochastic state evolution trajectories. If states cannot be directly observed, $p(t_E|o_{0:t_\text{p}})$ is computed. As defined, prognostics is only meaningful in a specific set of circumstances; we use two examples next to illustrate why this is so.

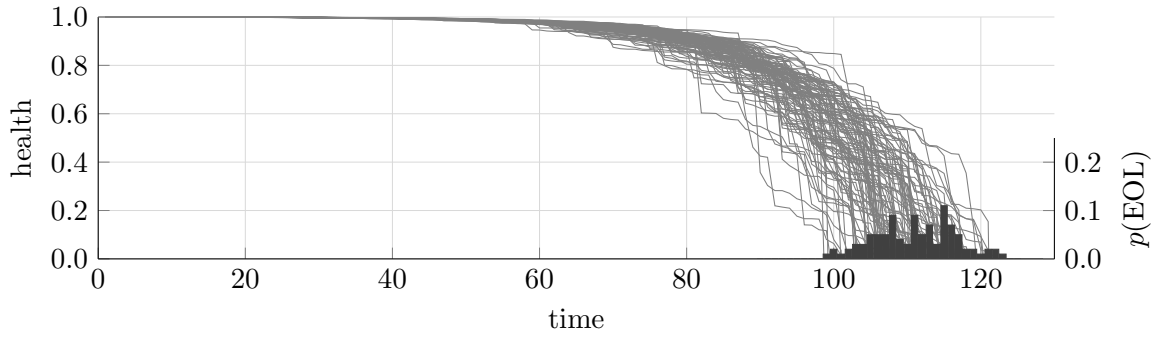

Figure 3: An EOL prediction illustration, with an overlay showing the $p(\text{EOL})$ distribution

4.1 Systems with Uncontrolled Degradation Processes

There are two main desirable, interrelated attributes for a prognostic module: (1) low uncertainty in EOL estimation, so that a decision about mitigation or recovery actions can be made with confidence, and (2) the ability to make a prediction far enough in advance that the actions can be successfully executed. In the case of uncontrolled degradation processes, this means that prognostics is primarily useful for those systems that have long expected lifetimes, low degradation process uncertainty, or both. To illustrate why this is the case, we start with a simple uncontrolled degradation example, shown here in Figure 4:

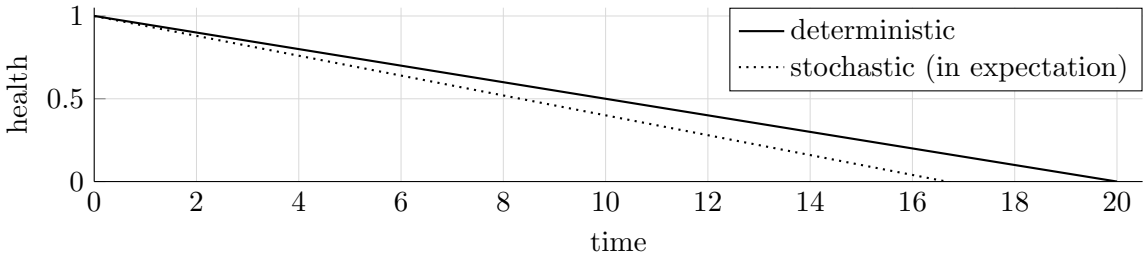

Figure 4: Prognostics of an uncontrolled degradation process with deterministic and stochastic models (Example 1)

Example 1. At time $t = 0$, the health state of a system is $s_0 = 1$ (states are scalar). According to a deterministic model, the nominal system health degradation rate is constant at $\dot{s}_\mathrm{n} = 0.05/\Delta t$, where Δt is the prediction time step, selected as the minimum time interval within which a change in the system's health is expected to be detectable. A stochastic model predicts the probability of the nominal degradation rate $\dot{s}_\mathrm{n}$ within any time step as $p_\mathrm{n} = 0.8$ and the probability of a higher degradation rate ($\dot{s}_\mathrm{h} = \dot{s}_\mathrm{n} + \epsilon/\Delta t$) as $p_\mathrm{h} = 0.2$. Assume $\epsilon = 0.05$.

The objective for both models is to predict EOL, i.e., the smallest t for which $s \leq 0$. For this example, the prediction uncertainty is defined as $\sigma(t_\mathrm{p}) = |E[\mathrm{EOL_d}(t_\mathrm{p})] - E[\mathrm{EOL_s}(t_\mathrm{p})]|$, i.e., the absolute difference between the expected EOL values computed by the two models at a prediction time t_p. A requirement is set on the maximum EOL prediction uncertainty as $\sigma_\mathrm{max} = 1\Delta t$.

For this example, let us assume that the health state is fully observable and define the fraction of full health remaining at t_p as $\rho_\mathrm{p} = s_\mathrm{p}/s_0$. In Figure 4, a prediction is shown to be made at $t_\mathrm{p} = t_0$, with $s_\mathrm{p} = s_0$, $\rho_\mathrm{p} = 1$, and the prediction horizon $H = 20\Delta t$. Since

$$E[\mathrm{EOL_d}(t_\mathrm{p})] = \frac{\rho_\mathrm{p}}{\dot{s}_\mathrm{n}} \tag{1}$$

and

$$E[\mathrm{EOL_s}(t_\mathrm{p})] = \frac{\rho_\mathrm{p}}{p_\mathrm{n}\dot{s}_\mathrm{n} + p_\mathrm{h}\dot{s}_\mathrm{h}} = \frac{\rho_\mathrm{p}}{(1-p_\mathrm{h})\dot{s}_\mathrm{n} + p_\mathrm{h}(\dot{s}_\mathrm{n} + \epsilon/\Delta t)} = \frac{\rho_\mathrm{p}}{(\dot{s}_\mathrm{n} + p_\mathrm{h}\epsilon/\Delta t))}, \tag{2}$$

then

$$\sigma = \left|\frac{\rho_\mathrm{p}}{\dot{s}_\mathrm{n}} - \frac{\rho_\mathrm{p}}{(\dot{s}_\mathrm{n} + p_\mathrm{h}\epsilon/\Delta t)}\right| = \rho_\mathrm{p}\left|20 - \frac{1}{(0.05 + p_\mathrm{h}\epsilon)}\right|\Delta t. \tag{3}$$

In the last equation, we substitute the value for the nominal degradation rate $\dot{s}_{\mathrm{n}}$ in order to focus on the effects of the degradation rate uncertainty. As can be seen in Figure 4, EOL is reached by both models within the prediction horizon. However, from Equation 3, $\sigma = 3.33\Delta t > \sigma_{\max}$.

For the requirement on σ to be satisfied, either ρ_{p} (health fraction remaining), p_{h} (the probability of deviations from the nominal degradation rate), ϵ (the magnitude of deviations), or some combination of them needs to be reduced. If p_{h} and ϵ are kept the same, with $\rho_{\mathrm{p}} = 0.25$ we can get $\sigma = 0.83\Delta t$. However, t_{p} now needs to be $\approx 15\Delta t$, with only $5\Delta t$ left until failure (i.e., RUL, the remaining useful life). RUL corresponds to the time available to either replace/repair the uncontrolled system or initiate an emergency response. For a quickly degrading system with $\Delta t = 1\,\mathrm{s}$, RUL would only be 5 s, which is likely enough time for an emergency response, but not for repair or replacement. In practice, in systems where responding to a fast-developing uncontrolled degradation process is important, estimating $p(\mathrm{EOL})$ is unlikely to bring tangible benefits. For instance, if pressure starts building quickly in a fuel tank of an ascending crewed rocket, the launch abort system (emergency response) is likely to be activated by the exceedance of a predefined pressure limit or pressure increase rate (i.e., functions of fault detection). Computing whether the tank breach will occur in 10 or 12 seconds will not materially influence that response. It may also be difficult to have high confidence in a prognostic prediction from a limited number of observations of a fast-developing degradation process.

There are some processes, mostly chemical in nature, with mid-range degradation rates (minutes, hours, days) that could be considered uncontrolled. One example of such a process is the decomposition of hydrogen peroxide, which is used as propellant in the attitude control thrusters of a Soyuz spacecraft descent module (Hall & Shayler, 2003). Over time, some of the hydrogen peroxide breaks down into water and oxygen, decreasing the amount of usable propellant. This limits the on-orbit life of the vehicle to about 200 days. For processes like this, extrapolation of the degradation function could be part of a caution and warning mechanism.

What follows from Example 1 is that if the degradation process uncertainty is relatively high or varies significantly over time, a short prediction horizon (compared to the overall system lifetime) may be necessary to limit the uncertainty propagation and result in a usable σ. In this case, systems with longer lifetimes are more suitable for applying prognostics. For example, if a bridge failure can be predicted three years in advance with an accuracy of ± 1 year, that can still be a useful prediction.

However, while many systems with uncontrolled degradation processes could be classified as having long expected lifetimes, there exists a number of fundamental practical difficulties in performing effective prognostics for them. One of the primary issues stems directly from the typically long lifetimes (often decades). In order to establish trust in the degradation models, they need to be adequately tested using long-term observations from "normal use" or observations from properly formulated accelerated testing. Useful and valid (from the statistical point of view) "normal use" degradation data sets are rare for most long-duration degradation processes (Eker, Camci, & Jennions, 2012; Heng, Zhang, Tan, & Mathew, 2009). Run-to-failure data sets collected in "normal use" conditions are particularly rare as, in practice, operating a system to complete failure (or even close to it) may have cost or safety implications (Eker et al.). Accelerated degradation efforts in controlled settings are,

therefore, quite common. If an accelerated degradation regime is proposed, however, what needs to be clearly demonstrated is that:

1. **The regime can be used as a substitute for real-life degradation processes**. For instance, while Rigamonti, Baraldi, Zio, Astigarraga, and Galarza (2016) and Celaya, Kulkarni, Biswas, and Saha (2011) use thermal and electrical overstress to quickly degrade electrical capacitors and predict their time of failure (by using empirical equations), their work does not make a connection to real-life degradation, which takes place at lower temperatures and voltage/current levels. Similar issues are highlighted by Dao, Hodgkin, Krstina, Mardel, and Tian (2006a, 2006b) for composite materials, where mechanical, thermal, and chemical processes result in complex interactions during aging.

2. **There is a known mapping from the accelerated timeline to the unaccelerated timeline.** To illustrate this requirement, we will again use electronic components as an example. In an overview of condition monitoring and prognostics of insulated gate bipolar transistors, Oh, Han, McCluskey, Han, and Youn (2015) note that numerous specialized fatigue models have been constructed that aim to predict *cycles-to-failure* under repetitive cycle loading in accelerated aging experiments. The application of various general fatigue analysis models, e.g., Coffin (1954), Miner (1945), or Matsuishi and Endo (1968), have also been proposed for predicting failure on the basis of usage cycles. As Oh et al. describe, however, there are significant practical challenges in using any of these methods to estimate a component's lifetime under realistic conditions and, as importantly, on realistic time scales. The first challenge is with accurately converting environmental and operational loading conditions into thermomechanical stresses. The second is with defining and extracting the number, amplitude, and duration of stress cycles when a system is subjected to a potentially complex usage profile. The third challenge is with properly accounting for damage accumulation and distribution in the various components of the system. In regard to the second challenge, Ciappa, Carbognani, Cova, and Fichtner (2003), for instance, proposed a method to decompose a complex usage profile into elementary cycles and compared it to the Coffin-Manson model (Coffin). The comparison was done on an accelerated aging data set only. While the proposed model performed better on the data set than the simpler Coffin-Manson model, the authors concluded that neither would be able to fully represent all of the mechanisms governing degradation under realistic use, including stress relaxation, anisotropic effects, and microstructural changes. The authors also note that even though more complicated models may be able to account for some of these additional mechanisms, the number of free parameters associated with such models would make their calibration virtually impossible.

There are subfields of prognostics where accelerated aging regimes may be viable, such as for metallic structures of aircraft or rotating machinery (where mechanical degradation factors could be assumed dominant). However, the issue of high uncertainty of degradation trajectories still arises, even for test articles made out of isotropic materials and aged under uniform conditions (Meng, 1994; Virkler, Hillberry, & Goel, 1979). Finite element modeling may help alleviate degradation trajectory uncertainty in specific cases, albeit at a significant

computational cost (Heng et al., 2009). Some of the other challenges preventing effective prognostics for systems with uncontrolled degradation include the accuracy of estimating the actual state of health, the effects of fault interactions, and the effects of system maintenance actions (Heng et al.).

If these challenges are successfully overcome and the failure mechanisms of a component are understood well enough to develop useful degradation models, a different question then arises: should the design or usage of the component be changed to mitigate these mechanisms? While in some cases this may not be feasible, in others it may be the simplest and most reliable way of improving safety and reducing system maintenance requirements (Bathias & Pineau, 2013). A redesign or change in usage would, on the other hand, make the degradation models obsolete. The next tier of degradation modes would then need to be analyzed and modeled, possibly followed by another redesign. Thus, analysis intended for the development of degradation (prognostics) models instead becomes part of the design improvement cycle.

In some instances, in addition to maintenance optimization, prognostics of long-lifespan components have been proposed as a means of ensuring safety of the overall system, even when the system's periods of active operational usage are substantially shorter than the expected lifetime of such components (e.g., Tang, Roemer, Bharadwaj, and Belcastro, 2008). To see why relying on prognostics for this function may not be advisable, consider the following example. A prognostic algorithm monitors the condition of an aircraft turbofan engine fan disk using a degradation model $\mathcal{M}$. A fan disk degradation Δt (the minimum time interval within which a change in component health is expected to be detectable) is measured in months, while a typical flight lasts several hours. If an in-flight failure of the fan disk is predicted to occur in, for example, 1 ± 0.25 hours, can $\mathcal{M}$ be trusted for determining a course of action, given that the impending failure was not forecast (and addressed) well before the current flight?

An argument can be made that even a healthy long-lifespan component can be damaged due to an unexpected (paroxysmal) event and projected to fail before the expected flight completion. This case is similar to the one described earlier (crewed rocket launcher), where a catastrophic short-duration degradation process takes place. Even assuming that an appropriate process model is available and that its parameters can be determined quickly and accurately, the optimal response will still likely be an emergency procedure (e.g., an emergency landing), prompted by an off-nominal state estimate rather than the prognostic prediction. If a coupling exists between emergency actions and the degradation process, then the case becomes that of a controllable degradation process, discussed in Section 4.2.

For systems with uncontrolled degradation processes that are, in fact, suitable for health management based on prognostics, the action space A is often limited to: (a) no action, (b) replacement, or (c) repair to achieve a nominal operating condition. Even so, we still propose that rather than following the sequence in Figure 2—i.e., computing $p(\text{EOL})$, then deriving decisions using that information—any predictive analysis should instead be driven by the requirements of a decision making procedure. This would allow for the formulation of an appropriate problem, development of suitable models, and determination of the maximum prediction (decision) horizon needed. For instance, if domain knowledge informs that variability in the system behavior past some health index $h_{\min}$ is too great for choosing actions with sufficiently high confidence (e.g., beyond $h = 0.8$ for the degradation process

depicted in Figure 3), then EOL can be redefined as $h_{\min}$ and system dynamics beyond $h_{\min}$ need to be neither modeled nor computed, potentially freeing computing resources for estimating system behavior up to $h_{\min}$ with more accuracy. In another scenario, assume that for some slow-degrading system the maximum execution time of any action in A (e.g., arranging for and performing system replacement) is $2\Delta t$. Then the question that needs to be answered at every decision step is not "When will the system fail?", but rather, "Will the system fail within the next $2\Delta t$?" The frequency of decision steps can then also be set to $2\Delta t$.

4.2 Systems with Controllable Degradation Processes

In a realistic controlled system, uncertainty is often present in state transitions. When the system's control actions can affect its degradation processes, prognostics, as defined at the beginning of Section 4, and the PHM version of the sequence depicted in Figure 2 are no longer meaningful—for two key reasons. First, not having the knowledge, at t_{p}, of the future system actions, a PHM algorithm will have to either (a) rely on some precomputed plan to obtain $a_{t_{\mathrm{p}}+1:H}$ for its predictive analysis (a plan that can quickly become obsolete due to action outcome uncertainty) or (b) use a random policy (which can, for instance, result in actions inappropriate for some system state being selected with the same probability as the more appropriate ones). A random policy is also likely to result in greater state uncertainty throughout the $[t_{\mathrm{p}}, t_{\mathrm{p}}+H]$ interval. Second, rerunning prognostic analysis after each action-initiated state transition to account for new information (e.g., Tang, Hettler, Zhang, and Decastro, 2011) may not always be helpful. Once a suboptimal execution branch has been committed to, it may remain suboptimal regardless of future decisions. The following example illustrates these issues:

> **Example 2.** Our lunar rover needs to traverse an area with no sunlight, going around a large crater from waypoint wp_0 to the closest suitable recharge location at wp_4 (Figure 5). For this example, we will consider the rover's battery state of charge to be its health indicator. At wp_0 the battery state of charge is 1100 Wh.

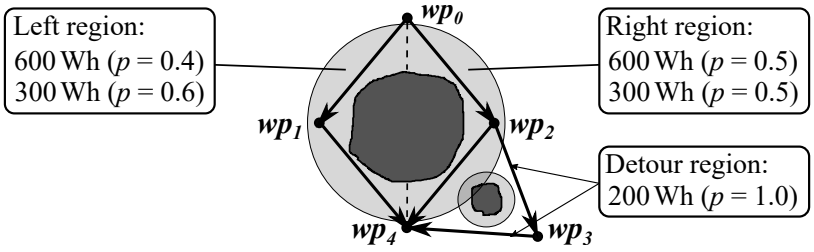

Figure 5: PHM vs. DM for a controlled system (Example 2)

There are three possible levels of terrain difficulty: *difficult* (requiring 600 Wh per drive segment), *moderate* (300 Wh per segment), and *easy* (200 Wh per segment). All drive segments are the same length. Probabilities of terrain types in different regions are shown in Figure 5.

The rover can go to the left, $wp_0 \to wp_1 \to wp_4$, or to the right, $wp_0 \to wp_2 \to wp_4$ (left and right are relative to the diagram). If going to the right, there is an

option to detour around a smaller crater $wp_2 \rightarrow wp_3 \rightarrow wp_4$ (*easy* terrain with $p = 1.0$) instead of going directly $wp_2 \rightarrow wp_4$.

A PHM algorithm used for decision support—running a sufficiently large number of simulations—would consider two possible execution scenarios along the left route: (L1) $e_{\text{total}} = 1200\,\text{Wh}$, $p = 0.4$ and (L2) $e_{\text{total}} = 600\,\text{Wh}$, $p = 0.6$ (e_{total} is the total energy consumed in a scenario). The expected energy consumption along the left route is, therefore, $E[e_{\text{total, L}}] = 1200\,\text{Wh} \cdot 0.4 + 600\,\text{Wh} \cdot 0.6 = 840\,\text{Wh}$ (Figure 6).

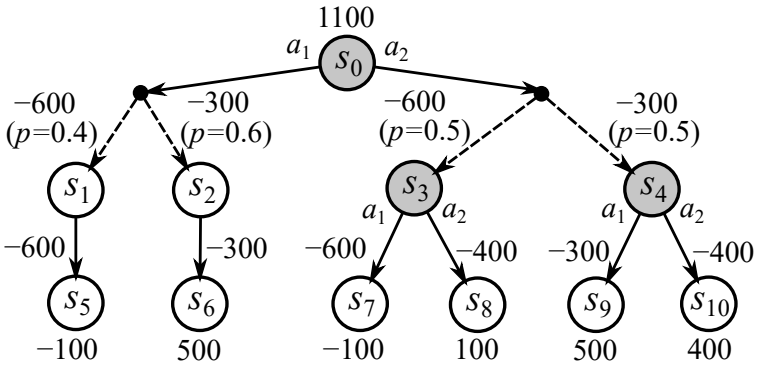

Figure 6: Execution scenarios in Example 2

The algorithm would then consider four possible execution scenarios along the right route (assuming uniform random choice of action at wp_3): (R1) $e_{\text{total}} = 1200\,\text{Wh}$, $p = 0.25$; (R2) $e_{\text{total}} = 600\,\text{Wh}$, $p = 0.25$; (R3) $e_{\text{total}} = 1000\,\text{Wh}$, $p = 0.25$; and (R4) $e_{\text{total}} = 700\,\text{Wh}$, $p = 0.25$. Then, $E[e_{\text{total, R}}] = (1200 + 600 + 1000 + 700) \cdot 0.25\,\text{Wh} = 875\,\text{Wh}$. With $E[e_{\text{total, L}}] < E[e_{\text{total, R}}]$, the PHM algorithm commits to the left path. Note that prognostics algorithms typically generate less information to support action selection than what was presented here, computing an aggregate $p(\text{EOL})$ only and not retaining potentially valuable performance data on individual execution trajectories (e.g., Daigle et al., 2015).

A DM algorithm capable of sequential reasoning under uncertainty (Kochenderfer, 2015) would compute $E[e_{\text{total, L}}] = 840\,\text{Wh}$ in the same manner as the PHM algorithm, as there are no actions needing to be selected along the left route after wp_0. On the right side, however, the DM algorithm can make an informed choice at wp_2, based on observations made along $wp_0 \rightarrow wp_2$. This means having only two possible execution scenarios: (R1) if the terrain is observed as *difficult*, the detour through wp_3 is taken, and (R2) if the terrain is observed as *moderate*, the rover goes directly to wp_4. For R1, $e_{\text{total}} = 1000\,\text{Wh}$, $p = 0.5$. For R2, $e_{\text{total}} = 600\,\text{Wh}$, $p = 0.5$. The expected energy use is thus $E[e_{\text{total, R}}] = (1000 + 600) \cdot 0.5\,\text{Wh} = 800\,\text{Wh}$. With $E[e_{\text{total, L}}] > E[e_{\text{total, R}}]$, the algorithm chooses the right path.

Now assume that the true terrain condition both on the left and right sides of the crater is *difficult*. The left path ($wp_0 \rightarrow wp_1 \rightarrow wp_4$) will require 1200 Wh to traverse, therefore a rover relying on the PHM algorithm will fall 100 Wh short and will not reach wp_4. A rover relying on the DM algorithm will expend only 1000 Wh (scenario R1), arriving at wp_4 with 100 Wh in reserve.

It may be suggested that the issues with the PHM approach could be eliminated if access to a precomputed operational policy π_{DM} is provided (rather than relying on some random policy). However, even if such a policy was accessible to PHM, that would still be insufficient. If, at time t, $p(\text{EOL})$ is computed using π_{DM}, then some $a_{t+1,\text{SHM}}$ is executed on the basis of $p(\text{EOL})$, $p(\text{EOL})$ would immediately become invalid unless $T(s_t, a_{t+1,\text{SHM}}, s_{t+1}) = T(s_t, a_{t+1,\text{DM}}, s_{t+1})$. In order to properly take $p(\text{EOL})$ into account, a new π_{DM} would need to be computed, which PHM is not capable of doing on its own.

While in this section we outlined the reasons why the *prognostic procedure* is not useful in most decision making situations, that does not mean that a *model* $\mathcal{M}$ of some relevant degradation process cannot be of value. Quite the contrary, if such a model is successfully developed—given the challenges described in Section 4.1—it can be an important part of the overall system state transition model. However, rather than computing $p(\text{EOL})$ or the probability distribution for time of occurrence of some other event E, $\mathcal{M}$ would need to be of the form $T(s, a, s')$, describing the probability of transitioning to a particular state s' as a result of action a. Further, $\mathcal{M}$ would need to be defined for $S \times A \times S$, i.e., for all valid combinations of system states (including environmental conditions, if so specified), actions, and follow-on states.

5. Unifying Decision Making and System Health Management

In this section we use additional numerical examples to illustrate the two main limitations of the current SHM approach: (1) the difficulty of balancing system health needs and operational objectives and (2) separated system models resulting in inferior solutions for both SHM and DM. The examples also show how the proposed unified approach would help to overcome these limitations. The key implementation details of the unified approach are then described, including the rationale for the more deliberative SHM/DM to be implemented separately from the more reactive emergency response capability (Section 5.3).

5.1 Separated vs. Unified SHM/DM

Our next example illustrates how unification can help balance frequently diverging system health needs and operational objectives:

> **Example 3.** The rover starts at wp_0 in Zone 1 (Figure 7) with 500 Wh (out of the overall 1500 Wh capacity). The solar panels can charge the battery at a rate of 250 W. Sunlight in Zone 1 will last for another 12 hours.

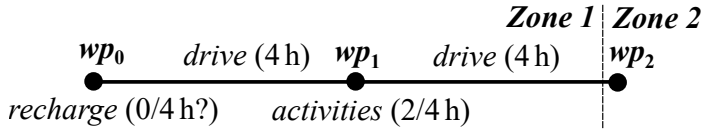

Figure 7: Balancing SHM and DM objectives (Example 3)

Two actions are available in the initial state s_0 at wp_0 (Figure 8): skip charging ($a_1 = +0$ Wh) and charge to full capacity ($a_2 = +1000$ Wh). In Figure 8, the unitless

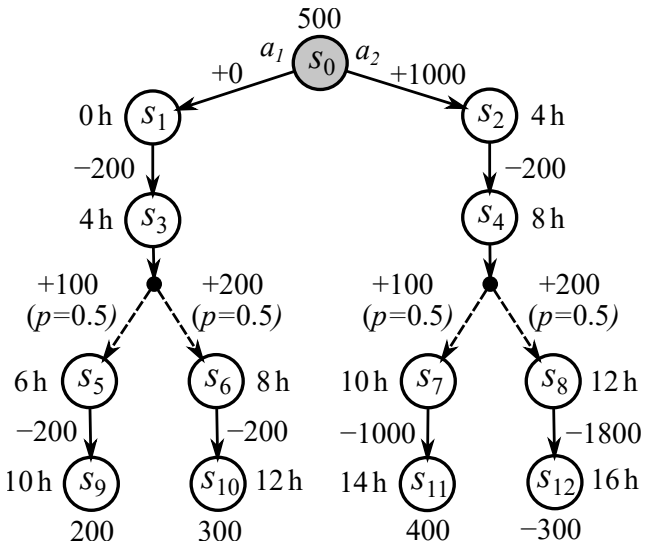

Figure 8: Execution scenarios in Example 3

numbers are Watt-hours of energy going in or out of the battery. Time (in hours) at each state is denoted as '[t] h'.

The rover needs to perform a two-hour stationary science activity at wp_1 and be able to arrive at wp_2, the next recharging point. The prior probability of the activity at wp_1 needing to be redone (a two-hour delay) is 0.5. The science payload power consumption is 100 W, resulting in a net-positive power flow (250 W − 200 W = +50 W).

Driving times from wp_0 to wp_1 and from wp_1 to wp_2 are 4 hours, with the average drive train power consumption of 300 W, resulting in a net-negative power flow (250 W − 300 W = −50 W). If operating without sunlight, a 150 W heater needs to be used to keep the batteries and electronics warm, thus resulting in a net-negative power flow of −300 W − 150 W = −450 W for driving and −200 W − 150 W = −350 W for stationary science activities.

According to the general SHM policy of restoring health (battery charge, in this case) to nominal, the action chosen at wp_0 is $\pi_{\text{SHM}}(s_0) = a_2$ and so the battery is recharged to its full capacity, 1500 Wh. After the science activity at w_1 is completed, an assessment is made that the activity needs to be repeated. The two-hour delay means that the entirety of the $wp_1 \to wp_2$ segment needs to be done without sunlight, resulting in complete battery depletion before wp_2 is reached (a deficit of 300 Wh).

In computing a unified policy, however, where SHM actions are considered in the context of the overall mission, all four scenarios depicted in Figure 8 would play a role. If a_1 is chosen, the expected amount of battery charge remaining would be $Q_1 = 0.5 \cdot 200\,\text{Wh} + 0.5 \cdot 300\,\text{Wh} = 250\,\text{Wh}$. For a_2: $Q_2 = 0.5 \cdot 400\,\text{Wh} + 0.5 \cdot (-300)\,\text{Wh} = 50\,\text{Wh}$. Action a_1 (no recharge) would be selected and, with the two-hour delay at wp_1, the rover would arrive at wp_2 with 300 Wh still remaining.

This example illustrates the first benefit of unifying SHM and DM: the ability to naturally take operational objectives and constraints into account when making a system health recovery decision. The next example illustrates how DM, on the other hand, can benefit from a unified action space and access to health-related models:

> **Example 4.** The rover is traveling from wp_0 to wp_1 (flat terrain) when a decision is made at point A to make a detour and attempt data collection at a scientifically valuable wp_2, requiring a six-hour climb up a steep hill (Figure 9). The one-hour data collection activity at wp_2 must be completed before the loss of illumination in 10 hours.
>
> After completing the two-hour climb up to point B (1/3 of the way up), it is observed that the internal temperature of one of the drive motors has risen to $T_{\mathrm{m}} = 60\,^{\circ}\mathrm{C}$ from the nominal $20\,^{\circ}\mathrm{C}$. At $T_{\mathrm{m}} = 80\,^{\circ}\mathrm{C}$, there is a significant risk of permanent damage and failure of the motor.

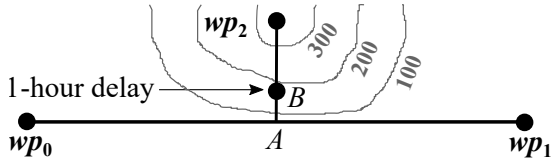

Figure 9: Separated versus unified SHM/DM models (Example 4)

If SHM on the rover consists of traditional fault detection, diagnosis, and mitigation/recovery components only, it may diagnose the fault to be *increased friction*, mark the component (motor) as faulty, and set a constraint on the acceptable terrain types to either *flat* or *downhill*. It would then invoke the recommended action for this fault from A_{SHM}: *stop and cool down* (until the motor temperature reaches $20\,^{\circ}\mathrm{C}$).

If a prognostic component is present, it may predict that, at the current temperature increase rate, the RUL of the motor is one hour (with four hours of climb remaining to reach wp_2). The same mitigation action (*stop and cool down*) would be initiated and the same constraint on the current to the motor (and, thus, on the incline angle) may be set. After T_{m} returns to nominal (which happens to take one hour), control is returned to DM. Given the new terrain type constraints and that the motor is classified as faulty, DM would command the rover to abort the detour, return to point A, and resume the drive to wp_2.

If, however, DM had *stop and cool down* as part of its action space and updated the state variables for the affected motor with the newly computed heat-up and cool down rates, an operational policy could be computed that optimizes the duration of driving and cool down intervals and allows the rover to reach wp_2 in time. For instance, if the rover drives for two hours, then stops for an hour to cool down the motor, it can still reach wp_2 in $2 + 1 + 2 + 1 + 2 = 8$ hours. With the science activity taking one hour, there would still be an hour in reserve before the loss of sunlight at wp_2.

The preceding example illustrates that if a system has an SHM-specific action set A_{SHM}, unifying it with A_{DM} (and doing the same with the state transition models) may allow for computation of operational policies infeasible in a separated SHM/DM architecture. In some cases of degraded system performance, this could mean the difference between

abandoning the remaining operational objectives and accomplishing at least some subset of them. The next subsection formalizes the unification approach and shows how the SHM concepts described in Section 3 are accommodated.

5.2 Unification Approach

In proposing the unified SHM/DM approach, we rely heavily on *utility theory* (Fishburn, 1970). We believe that the following are the key ingredients for a successful unification: (1) a state-based system modeling framework and (2) a utility (value) function describing operational preferences for the system. A *utility function* U captures, numerically, the preferences over the space of possible outcomes. For a strictly deterministic system operating in a deterministic environment, where a plan $a_{0:H}$ up to a horizon H can be provided ahead of time, the utility of state s relative to the plan is:

$$U^{a_{0:H}}(s) = \sum_{i=0}^{H} R(s_i, a_i). \tag{4}$$

For systems with action outcome uncertainty, the expected utility associated with executing a policy π for t steps from state s can be computed recursively as

$$U_t^{\pi}(s) = R\left(s, \pi(s)\right) + \gamma \sum_{s'} T(s, \pi(s), s') U_{t-1}^{\pi}(s'), \tag{5}$$

where $\gamma \in [0, 1]$ is the discount factor that is sometimes used to bias towards earlier rewards, particularly in *infinite horizon* problems ($t \to \infty$). The optimal utility for a state can then also be computed recursively using

$$U_t^*(s) = \max_{a \in A} \left(R(s, a) + \gamma \sum_{s'} T(s, a, s') U_{t-1}^*(s') \right), \tag{6}$$

which for $t \to \infty$ becomes the *Bellman equation* (Bellman, 1957). Knowing the optimal utility function, we can derive an optimal policy:

$$\pi^*(s) = \arg\max_{a \in A} \left(R(s, a) + \gamma \sum_{s'} T(s, a, s') U^*(s') \right). \tag{7}$$

In problems with state uncertainty, beliefs $b \in B$ can take the place of states in equations 5–7 (Kaelbling et al., 1998). The *Markov property* is assumed, meaning that $T(s, a, s')$ does not depend on the sequence of transitions that led to s (Kemeny & Snell, 1983).

Now that the foundational concepts have been reviewed, we will focus on those most essential to the proposed unification: states and the reward function $R(s, a)$. States can be vector quantities (Section 2). For real-world problems, relevant elements of the operating environment are sometimes included in the state vector, either explicitly or implicitly (Ragi & Chong, 2014). For instance, the lunar rover state vector would certainly need to include the rover's x and y coordinates, but may also include time t. These three elements allow us to indirectly access other information about the environment, e.g., solar illumination, ambient temperature, communications coverage, and terrain properties.

Similarly, health-related elements can be included in the same state vector. For the rover, the battery charge would likely be in the state vector already for operational purposes. Adding battery temperature, however, would allow for better reasoning about the state of battery health when combined with information on ambient temperature, terrain, and recharge current. Thus, including even a few health-related elements in the state vector (already containing information about the environment and the general state of the system) can have a multiplicative effect on the amount of information available. The resulting size of the state vector may also end up being smaller than the union of separately maintained SHM and DM state vectors, as redundant elements are eliminated.

The reward function $R(s, a)$ encodes the costs and the rewards of being in a particular state (or taking a particular action in a state) and can be used, through the utility function, to induce the desired behavior. For many realistic problems, the reward function needs to combine costs or rewards associated with different state components. Several approaches have been proposed (Keeney & Raiffa, 1993), with additive decomposition being an effective option in many cases. The key property of the function is that by mapping multiple variables to a single number it allows us to compute $U(s)$ or $U(s, a)$ and translate a potentially complex DM formulation into an abstract utility maximization problem.

One notable consequence of health-related components being integrated into a common state vector is that, from the computational point of view, the concepts of fault and failure become somewhat superfluous. If subsets $S_{\text{fault}} \subset S$ or $S_{\text{failure}} \subset S$ are defined for the system, the framework described above will not do anything differently for them. The only essential subset of S is S_{T} (the terminal states). Failure states may be part of S_{T} if they result in termination of system operations; however, goal (success states) are members of S_{T} also. The only difference between them is in their $U(s)$ values. As long as a component fault or a failure does not lead to a transition to a terminal state, actions that maximize the expected value of that state will be selected—which, as it happens, implements the "fail operational" philosophy (National Aeronautics and Space Administration, 2012).

We will refer to this unified approach as *health-aware decision making (HADM)*. The rest of the major SHM concepts are incorporated into the new approach as follows. Fault detection and diagnostics are subsumed in belief estimation and updating, although these operations are, of course, used for nominal belief states as well. Uncertainty management can now be purposefully incorporated into the decision-making process by augmenting A with information gathering actions (Bonet & Geffner, 2000; Levesque, 1996; Spaan, Veiga, & Lima, 2015; Weld, Anderson, & Smith, 1998), evaluated and selected in the same context as other types of actions. For actively controlled systems, predictive simulations are simply an integral part of $U(s)$ calculation, where $T(s, a, s')$ serves as a one-step "prognostic" function (with degradation models, if any). Whereas prognostic algorithms applied to systems with controllable degradation are limited in their predictive ability due to the lack of knowledge about future actions, here the $U(s)$ calculation process is an exploration of possible execution scenarios. It thus combines s' or b' estimation with sequential action selection.

Figure 10 shows the overall HADM operational loop (assuming state and action outcome uncertainty). Once the initial belief b_0 is estimated at time t_0, either an offline π^* is referenced or an online π^* is computed to determine a_0^* (best action). The action is executed by the plant, transitioning to a new (hidden) state, and generating an observation o_0. The

observation is then used to update b_0 (typically through some form of Bayesian updating) and the process repeats until a terminal state is believed to be reached.

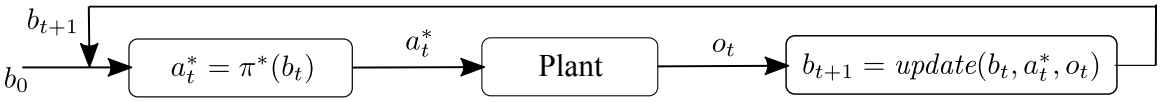

Figure 10: The main loop of health-aware decision making

A variety of algorithms capable of handling state or outcome uncertainty can be used to implement the proposed approach (Browne & Powley, 2012; Kochenderfer, 2015). In systems where both state and outcome uncertainty are not a factor, traditional state space planning algorithms (Ghallab et al., 2016) would not need any modifications to produce plans for spaces of state vectors that include health-related components.

In our discussion up to this point, we assumed that optimization of solutions is desired. If that is not the case, HADM can also be formulated from the satisficing perspective, e.g., as a constraint satisfaction problem (CSP). CSP formulation could be done using the same mathematical framework, with the problem solved by applying existing CSP algorithms (Frank, Jónsson, & Morris, 2000; Ghallab et al., 2016). In a CSP formulation, hard constraints could be defined by assigning a reward of $-\infty$ to undesirable states and soft (weighted) constraints could be defined using finite negative rewards, with all the other rewards set to zero.

5.3 Emergency Response versus Health-Aware Decision Making

For realistic complex systems operating in the presence of state and outcome uncertainty, S, B, O, and A are often high-dimensional and continuously valued. The problem of finding exact optimal policies in such cases is PSPACE-complete (Papadimitriou & Tsitsiklis, 1987). Approximate solution methods typically work online, constructing π for the current belief b (or a fully observable state s) based on beliefs/states reachable from b within the decision horizon (Browne & Powley, 2012). They also typically perform targeted sampling from S/B, A, and O, thus guarantees of true optimality can be harder to provide. We, therefore, propose

- That *system emergency response (SER)* be defined as an automated or semi-automated process that is invoked to maximize the likelihood of preserving the system's integrity, regardless of the effect on operational goals.
- That emergency response policy π_{SER} be computed separately from π_{HADM}.

In the space domain, an example of SER is commanding a spacecraft into *safe mode* until the emergency is resolved (Rasmussen, 2008). In aviation, it could be executing recommendations of a collision avoidance system (Kochenderfer, Holland, & Chryssanthacopoulos, 2013) or performing an emergency landing. Introduction of a separate SER system would likely require the introduction of S_{SER}, an additional subset of S that defines the states where π_{SER} is invoked versus π_{HADM}. Once again, however, S_{SER} will not necessarily only contain system fault and failure states. For instance, states where environmental conditions warrant emergency response (e.g., solar activity interrupting teleoperation of the rover)

would be included as well. While from a system architecture point of view SER may bear some resemblance to the SHM module in Figure 1, there are two important distinctions. First, SER would not operate in parallel with primary decision making, avoiding the potential for conflicting actions. Second, rather than being tasked with returning the system to a "healthy" state, SER is meant to merely transition the system to a state not in S_{SER}, then return control to HADM. Since the scope of the SER problem is likely to be much narrower than that of HADM, it opens the possibility of computing, verifying, and validating π_{SER} offline (Kochenderfer, 2015).

As sensors, computing capabilities, and DM algorithms improve, the fraction of the system's state space that is under the purview of SER will decrease. Still, we foresee the need for an independent, "safety net" SER to be there for safety-critical functions. SER would cover situations where the primary HADM system may not be able to produce a suitable solution in a desired amount of time, serving as an equivalent of human reflexive responses triggered by important stimuli versus the more deliberative cognitive functionality of the brain. It could also provide dissimilar redundancy for critical regions of S, essentially implementing the Swiss Cheese Model (Reason, 1990). In the model, multiple different layers of protection for important system functions are proposed, illustrated as slices of cheese. The holes in the slices, representing weaknesses in individual layers, vary in size and position from slice to slice (and, also, often vary in time). A system failure may occur if holes in all the protective layers happen to momentarily align. A greater number of dissimilar protective layers can, consequently, reduce the probability of failure.

6. Computational Complexity of Health-Aware Decision Making

A potential implementation issue for the proposed unified approach is that it may result in an increased computational complexity by operating on state, belief, observation, and action spaces that are higher-dimensional or larger (i.e., contain an increased number of elements in the same dimensions) compared with a separated SHM/DM formulation. In this section we examine the implications of S/B, O, and A dimensionality or size increase for problems with different types of uncertainties present. We will generally analyze the worst-case scenario (from a computational complexity point of view) by assuming that all spaces are continuously valued and, therefore, contain an infinite number of elements per dimension. We further assume that optimality (or near-optimality) is desired for both SHM and DM, therefore formulating both separated and unified SHM/DM cases as search problems. The last assumption also implies that computed decisions are not limited to being single-step reactive, but can be sequential.

We assume that the number of independent dimensions may increase for unified state, belief, and observation spaces, but will remain the same for unified action spaces. To see why, let us first consider a lunar rover example. Assume that in a separated SHM/DM implementation, a single action, originating either from SHM or DM, is chosen at each decision interval. The DM action space may contain a *drive* command, parameterized with real-valued heading and speed, thus forming its own class of actions with an infinite number of elements. The SHM action space may contain a *cool down* command, parameterized with real-valued duration, forming another infinitely large class of actions. Merging the two action spaces will not introduce additional dimensions, as at each time step only a single

action would still be selected (one of the *drive* options, for instance, or a *cool down* of a particular duration). Multi-dimensional action spaces are typically associated with multi-agent scenarios formulated as a single problem or, perhaps, with scenarios where multiple distinct components of a complex system need to be commanded simultaneously (e.g., a robotic manipulator with multiple independently commanded joints). Including health-related actions would not increase the number of dimensions in most of these cases either, as health-related action classes would be added to already existing dimensions.

6.1 Fully Deterministic Problems

If all sources of uncertainty can be neglected, classical search algorithms, such as Depth First Search, Breadth First Search, or A^* (Ghallab et al., 2016), can be applied to compute exact plans in problems of suitable size. The worst-case computational complexity will be $\mathcal{O}(|A_{\text{SHM}}|^D + |A_{\text{DM}}|^D)$ for the separated formulation and $\mathcal{O}(|A_{\text{SHM}} \cup A_{\text{DM}}|^D)$ for the unified one, with an overall worst-case computational complexity increase factor $\mathcal{O}\left((|A_{\text{SHM}} \cup A_{\text{DM}}|^D)/(|A_{\text{SHM}}|^D + |A_{\text{DM}}|^D)\right)$. Approximate algorithms, including those with the *anytime* property, can be used to trade optimality guarantees for performance in larger problems (Boddy, 1991; Burfoot, Pineau, & Dudek, 2006; Hansen & Zhou, 2007). If a problem formulation includes a continuously valued action space, it may need to be sampled; in that case, $|A|$ is the number of samples.

6.2 Problems with Action Outcome Uncertainty

Problems that incorporate action outcome uncertainty are often modeled as Markov decision processes (MDPs) (Bellman, 1957). Exact algorithms for generating optimal MDP policies include value iteration (Bellman) and policy iteration (Howard, 1960). Value iteration has complexity $O(|A||S|^2)$ per iteration; however, the number of iterations may grow exponentially in the MDP discount factor (Condon, 1992; Kaelbling, Littman, & Moore, 1996). Policy iteration has a significantly higher per iteration cost, $O(|A||S|^2 + |S|^3)$, although in practice may converge in fewer iterations than value iteration (Kaelbling et al., 1996; Littman, Dean, & Kaelbling, 1995). Methods for computing approximately optimal policies for larger problems include sparse sampling (Kearns, Mansour, & Ng, 2002), LAO^* (Hansen & Zilberstein, 2001), and Monte Carlo Tree Search (Browne & Powley, 2012). The latter has become widely used in recent years for MDP applications due to its scalability to large state spaces and its *anytime* property (a valid solution is returned even if the algorithm is interrupted, with solution quality improving as more iterations of the algorithm are executed). We will use Monte Carlo Tree Search (MCTS) in our analysis.

Section 4.2 showed that for an actively controlled system operating in the presence of uncertainty, informing SHM mitigation or recovery action selection with prognostic predictions can be ineffective. Therefore, rather than estimating the computational complexity of SHM implemented as the sequence in Figure 2, we assume that both SHM and DM problems are formulated as MDPs. The computational budget of forward simulations used by prognostics to construct a $p(\text{RUL})$ distribution can instead be allocated towards building an online MCTS policy, for instance.

In MCTS, as described by Browne and Powley (2012), Monte Carlo simulations from an initial state s_0 are used to build a partial search tree and, ultimately, estimate the expected

utility of each action available in s_0 (as MCTS is tree-based, continuously valued action spaces would need to be sampled). Action selection during tree traversal is typically done with one of the Upper Confidence Bound (UCB) algorithm variants (Auer, Cesa-Bianchi, & Fischer, 2002). The error in approximating the optimal policy depends on how many simulations passed through each node of the tree. For simplicity, we assume that the action space size $|A|$ is the same for all states and that each valid action is being invoked exactly N times on every level of the tree, up to a depth D. Given these assumptions, $N^D|A_{\text{SHM}} \cup A_{\text{DM}}|^D$ simulations would be required for a tree of depth D in the unified case versus $N^D(|A_{\text{SHM}}|^D + |A_{\text{DM}}|^D)$ in the separated case. Thus, we can estimate the computational complexity increase for the unified case to be $\mathcal{O}\left((|A_{\text{SHM}} \cup A_{\text{DM}}|^D)/(|A_{\text{SHM}}|^D + |A_{\text{DM}}|^D)\right)$.

6.3 Problems with State and Outcome Uncertainty

A generalization of an MDP, a partially observable Markov decision process (POMDP) is currently the prevalent mathematical framework for acting under both action outcome and state estimation uncertainty (partial observability). We will use POMDPs for our analysis of this final (and most complex) case. As mentioned in Section 5.3, computing optimal policies exactly under action outcome and state estimation uncertainty is generally considered a PSPACE-complete problem. For small discrete POMDPs, exact optimal policies can be computed (Kochenderfer, 2015). In most cases, however, approximately optimal policies are derived using either offline or online algorithms (Kurniawati, Hsu, & Lee, 2008; Pineau, Gordon, & Thrun, 2006; Ross, Pineau, & Paquet, 2008).

Analogously to the MDP case, we assume that both SHM and DM problems are formulated as POMDPs. In our analysis we will use two popular online POMDP solvers based on Monte Carlo sampling: Partially Observable Monte Carlo Planning (Silver & Veness, 2010) and Determinized Sparse Partially Observable Tree (Bai, Cai, Ye, Hsu, & Lee, 2015; Ye, Somani, Hsu, & Lee, 2017). We first examine the effects of unified A and O spaces on the accuracy of approximating an optimal policy, then discuss the implications of a unified, potentially higher-dimensional S on belief approximation and updating.

The Partially Observable Monte Carlo Planning (POMCP) algorithm is a straightforward extension of MCTS for partially observable domains. Instead of a partial search tree of states, POMCP builds a partial search tree of histories. Each node in the tree (corresponding to a unique history h) contains a set of unweighted *particles* (Del Moral, 1996), forming an approximate belief $\hat{b}_h$. Similarly to MCTS, the error in approximating the optimal policy will depend on the number of simulations that passed through each history (belief) node of the tree. Unlike MCTS trees, however, levels of history nodes in POMCP trees are interleaved with levels of observation nodes, resulting in $\mathcal{O}(|A|^D|O|^D)$ history nodes for a planning horizon of length D. For continuously valued action spaces, $|A|$ is the number of sampled actions and for simplicity we assume, again, that enough simulations are executed to generate $|A|$ branches out of every history node. We also assume that $|O|$ is the number of branches under each observation node. For continuously valued multidimensional observation spaces a strategy to add branches that results in a sufficient representation of the entire observation space may be required. Each simulation executed on a partially constructed tree adds exactly one new history node (Silver & Veness, 2010), therefore $\mathcal{O}(|A|^D|O|^D)$ simulations are required to construct the tree.

In a unified formulation, we may need $\mathcal{O}(|A_{\text{DM}} \cup A_{\text{SHM}}|)$ actions to represent the combined action space. The size of O_{HADM}, however, may need to be larger than $|O_{\text{DM}} \cup O_{\text{SHM}}|$ in order to adequately represent a higher-dimensional combined observation space. Thus we can estimate the POMCP computational complexity increase factor for the unified formulation as $\mathcal{O}\left(|O_{\text{HADM}}|^D |A_{\text{SHM}} \cup A_{\text{DM}}|^D / (|O_{\text{SHM}}|^D |A_{\text{SHM}}|^D + |O_{\text{DM}}|^D |A_{\text{DM}}|^D)\right)$.

Due to the properties of UCB-style action selection, the worst-case running time of POMCP is rather poor: $\Omega(\exp(\exp(\dots \exp(1) \dots)))$, nested $D-1$ times (Coquelin & Munos, 2007). The Determinized Sparse Partially Observable Tree (DESPOT) algorithm avoids this issue by relying on a set of K scenarios sampled a priori to construct policies for the current belief state. A DESPOT scenario for a belief b is an infinite abstract random sequence:

$$\phi = (s_0, \phi_1, \phi_2, \dots), \tag{8}$$

where s_0 is a scenario starting state sampled according to b and ϕ_i is a real number sampled independently and uniformly from $[0, 1]$. The K start states of the scenarios form the approximate belief $\hat{b}_0$ (in the Anytime Regularized implementation of DESPOT, vectors of weighted particles are used to form the belief nodes of the tree). Numbers ϕ_i are used in a generative deterministic model $g(s, a, \phi)$ to produce next state-observation pairs (s', o'). When the model is simulated for an action sequence $(a_1, a_2, \dots)$ under a scenario $(s_0, \phi_1, \phi_2, \dots)$, it generates a simulation trajectory $(s_0, a_1, s_1, o_1, a_2, s_2, o_2, \dots)$. The simulation trajectory traces out a path $(a_1, o_1, a_2, o_2, \dots)$ from the root of the tree. All of the nodes and edges on this path are added to the tree. Each belief node b in the tree contains a set Φ_b of all scenarios it encounters. Repeating this process for *every* action sequence under *every* sampled scenario completes the construction of the tree. A standard belief tree of height D would have $\mathcal{O}(|A|^D |O|^D)$ nodes, while a corresponding DESPOT has $\mathcal{O}(|A|^D K)$ nodes for $|A| > 2$ because of the reduced observation branching.

Formally, a DESPOT policy π is a policy tree derived from a DESPOT $\mathcal{D}$. A policy tree has the same root as $\mathcal{D}$, but only retains one action branch at each internal belief node (all of the observation branches are retained, however). The set $\Pi_{b_0,D,K}$ consists of all policies derived from DESPOTs of height D, constructed with all possible K sampled scenarios for a belief b_0.

Ye et al. (2017) provide two useful theoretical results for the DESPOT algorithm. The first result bounds the error of estimating the value of any policy in $\Pi_{b_0,D,K}$, with the implication that a DESPOT constructed with a small number of scenarios is sufficient for approximate policy evaluation. The second shows that by optimizing this bound, a policy can be obtained that is competitive with the best small policy. The size $|\pi|$ of a DESPOT policy π is the number of belief nodes in its policy tree. Constraining policy size is important to prevent overfitting, as a policy optimized for a finite number of sampled scenarios may not perform well in general.

In proving the first result, Ye et al. (2017) show that for any given constants $\tau, \alpha \in (0, 1)$, any belief b_0, and any positive integers D (tree depth) and K, every DESPOT policy tree $\pi \in \Pi_{b_0,D,K}$ satisfies

$$V_\pi(b_0) \geq \frac{1-\alpha}{1+\alpha} \hat{V}_\pi(b_0) - \frac{R_{\max}}{(1+\alpha)(1-\gamma)} \cdot \frac{\ln(4/\tau) + |\pi| \ln(KD|A||O|)}{\alpha K} \tag{9}$$

with probability of at least $1-\tau$, where $\hat{V}_\pi(b_0)$ is the estimated value of π under a set of K scenarios randomly sampled according to b_0, $R_{\max}$ is the maximum reward, and γ is the POMDP discount factor. This implies that all DESPOT policies in $\Pi_{b_0,D,K}$ satisfy the bound given in (9) with high probability. The policy estimation error (second term on the RHS) can be made arbitrarily small by choosing an appropriate K.

The second result shows that a near-optimal policy $\hat{\pi}$ can be obtained by maximizing the RHS of (9). Let Π_D be the set of all policies derived from a DESPOT $\mathcal{D}$ with height D, constructed with K scenarios sampled randomly according to belief b_0. For any arbitrary policy $\pi \in \Pi_D$ and any given constants $\tau, \alpha \in (0,1)$, if

$$\hat{\pi} = \arg\max_{\pi' \in \Pi_D} \left\{ \frac{1-\alpha}{1+\alpha} \hat{V}_{\pi'}(b_0) - \frac{R_{\max}}{(1+\alpha)(1-\gamma)} \cdot \frac{|\pi'| \ln(KD|A||O|)}{\alpha K} \right\}, \tag{10}$$

then

$$\begin{aligned} V_{\hat{\pi}}(b_0) \geq & \frac{1-\alpha}{1+\alpha} V_\pi(b_0) - \frac{R_{\max}}{(1+\alpha)(1-\gamma)} \times \\ & \times \left(\frac{\ln(8/\tau) + |\pi| \ln(KD|A||O|)}{\alpha K} + (1-\alpha) \left(\sqrt{\frac{2\ln(2/\tau)}{K}} + \gamma^D \right) \right) \end{aligned} \tag{11}$$

with probability of at least $1-\tau$.

In expression (11), performance of $\hat{\pi}$—a policy maximizing (10)—is bounded relative to the performance of another policy, π. Since π can be any policy in Π_D, we can choose it to be an optimal policy π^*. If $|\pi^*|$ is small, the approximation error of $\hat{\pi}$ is also small. If π^* is large, but is approximated well by some small policy π of size $|\pi|$, then $\hat{\pi}$ can be obtained with a small approximation error by choosing K to be $\mathcal{O}(|\pi| \ln(KD|A||O|)$.

Let us consider how the RHS of (11) may change in a unified HADM formulation. Since we expect an optimal HADM policy to perform at least as well as either SHM or DM policy, $V_\pi(b_0)$—i.e., $V_{\pi^*}(b_0)$—will increase or remain the same. Rewriting the additive error on the RHS of (11) as

$$\epsilon = \frac{R_{\max}}{(1+\alpha)(1-\gamma)} \left(\frac{\ln(8/\tau) + |\pi^*| \ln(KD|A||O|)}{\alpha K} + (1-\alpha) \left(\sqrt{\frac{2\ln(2/\tau)}{K}} + \gamma^D \right) \right), \tag{12}$$

we will now estimate the ratio $K_{\text{HADM}}/(K_{\text{SHM}} + K_{\text{DM}})$, assuming that approximately the same ϵ is to be maintained for the unified HADM policy as for the separated SHM/DM policies. We assume that α, γ, τ, D, and $R_{\max}$ are the same for HADM as for SHM/DM. The $R_{\max}/\big((1+\alpha)(1-\gamma)\big)$ multiplier will, consequently, also remain the same. The second term inside the parentheses is small for realistic values of K (i.e., hundreds). The numerator of the first term is dominated by $|\pi^*|$ (the size of an optimal policy). We can, therefore, estimate $K_{\text{HADM}}/(K_{\text{SHM}}+K_{\text{DM}})$ to be $\mathcal{O}\big(|\pi^*_{\text{HADM}}| \,/(|\pi^*_{\text{SHM}}|+|\pi^*_{\text{DM}}|)\big)$. Just as for POMCP, the scenarios would have to be executed for a potentially larger number of actions on every level of the tree, thus the overall increase in DESPOT computational complexity due to unified O and A spaces will be $\mathcal{O}\big(|\pi^*_{\text{HADM}}| \,|A_{\text{SHM}} \cup A_{\text{DM}}|^D \,/(|\pi^*_{\text{SHM}}| \,|A_{\text{SHM}}|^D + |\pi^*_{\text{DM}}| \,|A_{\text{DM}}|^D)\big)$. An interesting implication of this result is that if a more compact optimal policy exists for HADM than for the separated SHM and DM (which may well be the case), the overall

DESPOT computational complexity may actually *decrease* (at least the complexity attributable to the effects of unified A and O spaces).

Next we focus on the potential effects of a higher-dimensional state space, which in some cases may present the greatest computational challenge. Both POMCP and DESPOT use particle-based belief representations and are typically coupled with a particle filtering algorithm (Gordon, Salmond, & Smith, 1993) to perform belief updating. Snyder, Bengtsson, Bickel, and Anderson (2008) and Bengtsson, Bickel, and Li (2008) point out the main issue with using particle filters in high-dimensional spaces: particle filters that use an insufficiently large set of particles in such spaces tend to *collapse*, with a single particle ending up with a weight close to unity. In order to maintain representational quality and prevent filter collapse, the number of particles in the set must grow exponentially with the number of space dimensions—or, more precisely, with the variance of the observation log likelihood, as Snyder et al. demonstrate. For use with an online POMDP solver, such as POMCP or DESPOT, particles at the root of a partial belief tree form both the current belief approximation and the start states of simulations/scenarios used to construct the tree. The number of POMCP simulations or DESPOT scenarios needed will then be the greater of the two: the number required to maintain the quality of the optimal policy approximation or the number required for an effective state space coverage.

Several approaches have been proposed for handling high-dimensional state spaces in belief representation and updating. Roy, Gordon, and Thrun (2005) observe that in real-world POMDP problems, computing an optimal policy for the entire belief space is often unnecessary and that the beliefs relevant for decision making often lie near a structured, low-dimensional subspace embedded in the high-dimensional belief space. They, therefore, propose reducing the dimensionality of the belief space (prior to policy computation) by using the exponential family principal components analysis. Bengtsson, Snyder, and Nychka (2003) describe a nonlinear ensemble filter that can handle non-Gaussian probability densities as an alternative to particle filters for high-dimensional systems. Importance sampling has also been suggested as a way to design particle filters that are scalable to high-dimensional problems (Daum & Huang, 2003). Luo, Bai, Hsu, and Lee (2019) implement a version of DESPOT that uses importance sampling in scenario selection, which, in particular, can be helpful for ensuring that the computed policy addresses rare events that otherwise would be difficult to sample. For HADM, this could mean the ability to capture policy responses to fault modes, especially those for which degradation models are not part of the overall state transition function $T(s, a, s')$. With all importance sampling approaches, constructing a good proposal distribution is key. In most applications this has been done manually, relying on domain knowledge. Luo et al. suggest a method for automatically constructing proposal distributions, specifically geared for POMDP policy computation.

Rebeschini and Van Handel (2015) provide a mathematical foundation for another approach to dealing with high-dimensional state spaces. The central idea of the approach is that the *decay of correlations* property, which could be viewed as a spatial counterpart of the *stability* property of nonlinear filters, can be exploited to develop *local* particle filters that scale well to high-dimensional settings. The authors also present an example algorithm, Block Particle Filtering, for which they prove an approximation error bound that is uniform in both time and dimension. Beskos, Crisan, Jasra, Kamatani, and Zhou (2017) extend a related idea into the Space-Time Particle Filter that combines local filtering along the space

dimensions with global filtering in the time dimension to provide sub-exponential computational complexity in the number of space dimensions. The authors provide theoretical and numerical results showing consistency of the filter and its stability in high dimensions for certain model classes.

6.4 Discussion

The analysis presented above is not exhaustive across all possible types of DM methods. Its objective, however, is to illustrate that practical methods for implementing unified HADM exist, and while there may be increases in computational complexity relative to separated implementations, such increases are not insurmountable even in the most complex cases.

Fully deterministic HADM problems and problems with action outcome uncertainty can be solved optimally or near-optimally with a computational complexity factor increase of, at worst, $\mathcal{O}\left(|A_{\text{DM}} \cup A_{\text{SHM}}|^D / (|A_{\text{DM}}|^D + |A_{\text{SHM}}|^D)\right)$ over a separated formulation (here D is equivalent to H, i.e., the planning horizon). While not negligible, such an increase should be manageable for the application domain being considered since $|A_{\text{SHM}}|$ is typically much smaller than $|A_{\text{DM}}|$, with most of the mitigation/recovery control accomplished through $|A_{\text{DM}}|$ (for many systems $A_{\text{SHM}} = \emptyset$, with A_{DM} used both for controlling the system and managing its health). Figure 11 shows how the computational complexity may increase for different decision horizon lengths if SHM-specific actions (A_{SHM}) constitute 5% and 10% of the unified action space A_{HADM}, respectively. One way to potentially alleviate an increase in the action space size is by using state-dependent action spaces, thus reducing the average action branching factor. Partially observable HADM problems could present a more challenging case from the computational complexity point of view. In addition to the complexity increase resulting from a larger action space, increases due to higher-dimensional state and observation spaces may play a role. We anticipate, however, that dimensionality increases for unified S and O will be moderated by elimination of redundant dimensions. Additionally, algorithms like DESPOT exist for this class of problems, with computational complexity depending primarily on the problem structure (size of the optimal policy) rather than on the dimensionality of S and O. It is quite possible that some HADM problems will have more compact optimal policies than their separated formulations by the virtue of having access to the unified model spaces.

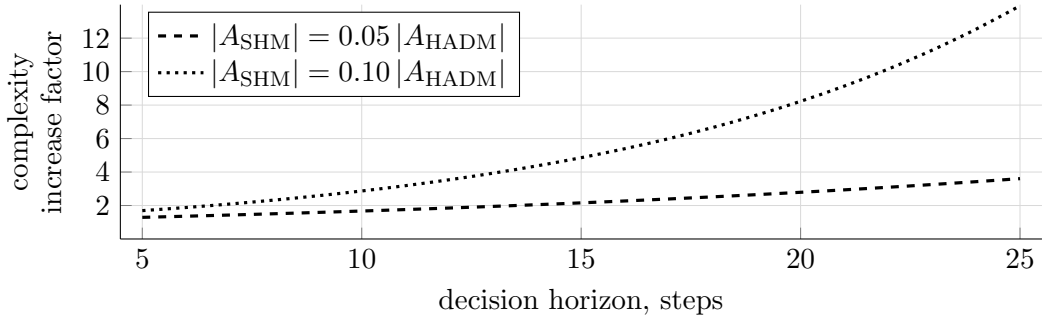

Figure 11: Examples of computational complexity increase for different decision horizons in fully deterministic problems and problems with action outcome uncertainty

Finally, many modern DM algorithms lend themselves well to parallelization. For instance, HyP-DESPOT (Cai, Luo, Hsu, & Lee, 2018) leverages massive parallelization on both CPUs and GPUs to demonstrate a speed-up factor of a few hundred times relative to the original DESPOT algorithm on several benchmark problems. We expect the emerging high-dimensional belief update methods referenced in the previous subsection to benefit from parallelization as well.

7. Concluding Remarks

This paper reexamines the prevalent approach to performing system health management and makes the case for why keeping system health management functions separate from decision making (planning, scheduling, and control) can be inefficient and ineffective. We also show why prognostic analysis of system health is only meaningful in a limited set of circumstances and, even then, needs to be driven by decision-making requirements.

We then explain the rationale for unifying—not just integrating—system health management with decision making, outline an approach for accomplishing unification, and show how traditional system health management concepts map into the new approach. We also propose keeping emergency response functionality separate to guarantee timely reactions, provide dissimilar redundancy, and allow for offline computation, validation, and verification of emergency response policies.

With the unified approach being suitable for systems of various complexity and types of uncertainties present, a wide range of existing and emerging computational methods can be used to implement it. While there may be an overall increase in computational cost under the proposed approach (as compared to a separated formulation), we believe that advances in algorithms and computing hardware generally make that a surmountable challenge.

In return, the approach is expected to improve a system's resilience to faults and its capacity to accomplish operational objectives, while potentially simplifying its informational architecture, reducing the sensor suite size, and combining modeling efforts. It also enables methodical exploration of design choices that affect both decision making and system health during operation. In that, the proposed approach advances towards the vision long held by the model-based AI community: to leverage the same set of models for different functions throughout a system's lifetime.

Acknowledgments

The authors gratefully acknowledge funding from NASA Ames Research Center in support of this work. They are thankful for the productive discussions with their colleagues at NASA Ames and Stanford University (particularly to Zachary Sunberg, Zongzhang Zhang, and Ritchie Lee), as well as for the valuable comments on the earlier version of the paper (presented at the 2018 AAAI Fall Symposium on Integrating Planning, Diagnosis, and Causal Reasoning) by Liljana Spirkovska, Indranil Roychoudhury, Gregory Dorais, and Mark "Mak" Roberts. The authors also express their appreciation for the thoughtful feedback by the anonymous AAAI and JAIR reviewers, as the paper has undoubtedly benefited from it. Finally, they would like to thank Greg Orzech (NASA Ames) for taking the time to proofread the manuscript.

References

Aaseng, G. B. (2001). Blueprint for an integrated vehicle health management system. In *IEEE Digital Avionics Systems Conference*.

Agha-mohammadi, A., Ure, N. K., How, J. P., & Vian, J. (2014). Health aware stochastic planning for persistent package delivery missions using quadrotors. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*.

Auer, P., Cesa-Bianchi, N., & Fischer, P. (2002). Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, *47*(2-3), 235–256.

Avizienis, A. (1976). Fault-tolerant systems. *IEEE Transactions on Computers*, *25*(12).

Bai, H., Cai, S., Ye, N., Hsu, D., & Lee, W. S. (2015). Intention-aware online POMDP planning for autonomous driving in a crowd. In *2015 IEEE International Conference on Robotics and Automation*.

Balaban, E., & Alonso, J. J. (2013). A modeling framework for prognostic decision making and its application to UAV mission planning. In *Annual Conference of the Prognostics and Health Management Society*, New Orleans, LA.

Balaban, E., Arnon, T., Shirley, M. H., Brisson, S. F., & Gao, A. (2018). A system health aware POMDP framework for planetary rover traverse evaluation and refinement. In *AIAA SciTech Forum*.

Bathias, C., & Pineau, A. (2013). *Fatigue of materials and structures: fundamentals*. Wiley.

Bellman, R. (1957). A Markovian decision process. *Journal of Mathematics and Mechanics*, (6), 679–684.

Benedettini, O., Baines, T., Lightfoot, H., & Greenough, R. (2009). State-of-the-art in integrated vehicle health management. *Journal of Aerospace Engineering*, *223*(2), 157–170.

Bengtsson, T., Bickel, P., & Li, B. (2008). Curse-of-dimensionality revisited: collapse of the particle filter in very large scale systems. *Probability and Statistics: Essays in Honor of David A. Freedman*, *2*, 316–334.

Bengtsson, T., Snyder, C., & Nychka, D. (2003). Toward a nonlinear ensemble filter for high-dimensional systems. *Journal of Geophysical Research*, *108*(D24), 8775–8785.

Bernard, D., Dorais, G., Gamble, E., Kanefsky, B., Kurien, J., Man, G., Millar, W., Muscettola, N., Nayak, P., Rajan, K., Rouquette, N., Smith, B., Taylor, W., & Tung, Y.-W. (1999). Spacecraft autonomy flight experience: the DS1 remote agent experiment. In *AIAA Space Technology Conference and Exposition*.

Beskos, A., Crisan, D., Jasra, A., Kamatani, K., & Zhou, Y. (2017). A stable particle filter for a class of high-dimensional state-space models. *Advances in Applied Probability*, *49*(1), 24–48.

Boddy, M. (1991). Anytime problem solving using dynamic programming. In *AAAI Conference on Artificial Inteligence*.

Bonet, B., & Geffner, H. (2000). Planning with incomplete information as heuristic search in belief space. In *International Conference on Artificial Intelligence Planning Systems*.

Browne, C., & Powley, E. (2012). A survey of Monte Carlo Tree Search methods. *IEEE Transactions on Computational Intelligence and AI in Games*, *4*(1), 1–49.

Burfoot, D., Pineau, J., & Dudek, G. (2006). RRT-Plan: a randomized algorithm for STRIPS planning. In *International Conference on Automated Planning and Scheduling*.

Cai, P., Luo, Y., Hsu, D., & Lee, W. S. (2018). HyP-DESPOT: a hybrid parallel algorithm for online planning under uncertainty. In *Robotics: Science and Systems Conference*.

Celaya, J., Kulkarni, C. S., Biswas, G., & Saha, S. (2011). A model-based prognostics methodology for electrolytic capacitors based on electrical overstress accelerated aging. In *Annual Conference of the Prognostics and Health Management Society*.

Chryssanthacopoulos, J. P., & Kochenderfer, M. J. (2012). Hazard alerting based on probabilistic models. *Journal of Guidance, Control, and Dynamics*, *35*(2), 442–450.

Ciappa, M., Carbognani, F., Cova, P., & Fichtner, W. (2003). Lifetime prediction and design of reliability tests for high-power devices in automotive applications. *IEEE International Reliability Physics Symposium Proceedings*, *3*(4), 523–528.

Coffin, L. (1954). A study of the effects of cyclic thermal stresses on a ductile metal. *Transactions of the American Society of Mechanical Engineers*, *76*, 931–950.

Condon, A. (1992). The complexity of stochastic games. *Information and Computation*, *96*(2), 203–224.

Coquelin, P.-A., & Munos, R. (2007). Bandit algorithms for tree search. In *Uncertainty in Artificial Intelligence Conference*.

Daigle, M. J., & Roychoudhury, I. (2010). Qualitative event-based diagnosis: case study on the Second International Diagnostic Competition. In *International Workshop on the Principles of Diagnosis*.

Daigle, M. J., Sankararaman, S., & Kulkarni, C. S. (2015). Stochastic prediction of remaining driving time and distance for a planetary rover. In *IEEE Aerospace Conference*.

Dao, B., Hodgkin, J., Krstina, J., Mardel, J., & Tian, W. (2006a). Accelerated aging versus realistic aging in aerospace composite materials. I. The chemistry of thermal aging in a low-temperature-cure epoxy composite. *Journal of Applied Polymer Science*, *102*, 4291–4303.

Dao, B., Hodgkin, J., Krstina, J., Mardel, J., & Tian, W. (2006b). Accelerated aging versus realistic aging in aerospace composite materials. II. Chemistry of thermal aging in a structural composite. *Journal of Applied Polymer Science*, *102*, 3221–3232.

Daum, F., & Huang, J. (2003). Curse of dimensionality and particle filters. In *IEEE Aerospace Conference*.

Del Moral, P. (1996). Nonlinear filtering: interacting particle resolution. *Markov Processes and Related Fields*, *2*(4), 555–581.

Eker, O. F., Camci, F., & Jennions, I. K. (2012). Major challenges in prognostics: study on benchmarking prognostics datasets. In *European Conference of the Prognostics and Health Management Society*.

Feldman, A., Provan, G., & Gemund, A. V. (2010). A model-based active testing approach to sequential diagnosis. *Journal of Artificial Intelligence Research*, *39*, 301–334.

Ferrell, B. (2000). Air vehicle prognostics and health management. In *IEEE Aerospace Conference*.

Figueroa, F., & Walker, M. G. (2018). Integrated system health management (ISHM) and autonomy. In *AIAA SciTech Forum*.

Fishburn, P. C. (1970). *Utility theory for decision making*. Research Analysis Corporation.

Frank, J., Jónsson, A., & Morris, P. (2000). On reformulating planning as dynamic constraint satisfaction. In *International Symposium on Abstraction, Reformulation, and Approximation*.

Ghallab, M., Nau, D., & Traverso, P. (2016). *Automated planning and acting*. Cambridge University Press.

Gordon, N., Salmond, D., & Smith, A. (1993). Novel approach to nonlinear/non-Gaussian Bayesian state estimation. *IEE Proceedings F (Radar and Signal Processing)*, *140*(2), 107–113.

Hall, R., & Shayler, D. (2003). *Soyuz: a universal spacecraft*. Springer.

Hansen, E. A., & Zhou, R. (2007). Anytime heuristic search. *Journal of Artificial Intelligence Research*, (28), 267–297.

Hansen, E. A., & Zilberstein, S. (2001). LAO*: a heuristic search algorithm that finds solutions with loops. *Artificial Intelligence*, *129*, 35–62.

Heng, A., Zhang, S., Tan, A. C. C., & Mathew, J. (2009). Rotating machinery prognostics: state of the art, challenges and opportunities. *Mechanical Systems and Signal Processing*, *23*(3), 724–739.

Howard, R. A. (1960). *Dynamic programming and Markov processes*. MIT Press.

Iverson, D. L. (2004). Inductive system health monitoring. In *International Conference on Artificial Intelligence*.

Johnson, S. B., & Day, J. C. (2010). Conceptual framework for a fault management design methodology. In *AIAA Infotech@Aerospace Conference*.

Johnson, S. B., & Day, J. C. (2011). System health management theory and design strategies. In *AIAA Infotech@Aerospace Conference*.

Kaelbling, L. P., Littman, M. L., & Cassandra, A. R. (1998). Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, *101*(1-2), 99–134.

Kaelbling, L. P., Littman, M. L., & Moore, A. W. (1996). Reinforcement learning: a survey. *Journal of Artificial Intelligence Research*, *4*, 237–285.

Kearns, M., Mansour, Y., & Ng, A. Y. (2002). A sparse sampling algorithm for near-optimal planning in large Markov decision processes. *Machine Learning*, *49*, 193–208.

Keeney, R. L., & Raiffa, H. (1993). *Decisions with multiple objectives: preferences and value tradeoffs*. Cambridge University Press.

Kemeny, J., & Snell, J. (1983). *Finite Markov chains*. Springer.

Kochenderfer, M. J. (2015). *Decision making under uncertainty: theory and application*. MIT Press.

Kochenderfer, M. J., Holland, J. E., & Chryssanthacopoulos, J. P. (2013). Next-generation airborne collision avoidance system. *Lincoln Laboratory Journal*, *19*(1), 17–33.

Korbicz, J., Koscielny, J. M., Kowalczuk, Z., & Cholewa, W. (2012). *Fault diagnosis: models, artificial intelligence, applications*. Springer.

Kurien, J., & Nayak, P. P. (2000). Back to the future for consistency-based trajectory tracking. In *AAAI National Conference on Artificial Intelligence*.

Kurniawati, H., Hsu, D., & Lee, W. (2008). SARSOP: efficient point-based POMDP planning by approximating optimally reachable belief spaces. In *Robotics: Science and Systems Conference*.

Levesque, H. J. (1996). What is planning in the presence of sensing? In *National Conference on Artificial Intelligence*.

Littman, M. L., Dean, T. L., & Kaelbling, L. P. (1995). On the complexity of solving Markov decision problems. In *Uncertainty in Artificial Intelligence Conference*.

Lopez, I., & Sarigul-Klijn, N. (2010). A review of uncertainty in flight vehicle structural damage monitoring, diagnosis and control: challenges and opportunities. *Progress in Aerospace Sciences*, *46*(7), 247–273.

Lu, K., & Saeks, R. (1979). Failure prediction for an on-line maintenance system in a Poisson shock environment. *IEEE Transactions on Systems, Man, and Cybernetics*, *9*(6), 356–362.

Luo, Y., Bai, H., Hsu, D., & Lee, W. S. (2019). Importance sampling for online planning under uncertainty. *The International Journal of Robotics Research*, *38*(2-3), 162–181.

Matsuishi, M., & Endo, T. (1968). Fatigue of metals subjected to varying stress. *Japan Society of Mechanical Engineers*, *68*(2), 37–40.

Meng, H.-C. (1994). *Wear modeling: evaluation and categorization of wear models* (Doctoral dissertation, University of Michigan).

Metodi, A., Stern, R., Kalech, M., & Codish, M. (2014). A novel SAT-based approach to model based diagnosis. *Journal of Artificial Intelligence Research*, *51*, 377–411.

Meyer, C. M., Fulton, C., Maul, W., Chicatelli, A., Cannon, H., Bajwa, A., Balaban, E., & Wong, E. (2003). Propulsion IVHM technology experiment overview. In *IEEE Aerospace Conference*.

Miner, M. (1945). Cumulative damage in fatigue. *Journal of Applied Mechanics*, *12*(3), A159–A164.

Narasimhan, S., & Brownston, L. (2007). HyDE – a general framework for stochastic and hybrid model-based diagnosis. In *International Workshop on the Principles of Diagnosis*.

National Aeronautics and Space Administration. (2012). *Fault management handbook*.

Nicewarner, K., & Dorais, G. (2006). Designing and validating an adjustably-autonomous free-flying intraspacecraft robot. In *AIAA Space Conference*.

Ogata, K. (2010). *Modern control engineering*. Pearson.

Oh, H., Han, B., McCluskey, P., Han, C., & Youn, B. D. (2015). Physics-of-failure, condition monitoring, and prognostics of insulated gate bipolar transistor modules: a review. *IEEE Transactions on Power Electronics*, *30*(5), 2413–2426.

Papadimitriou, C., & Tsitsiklis, J. (1987). The complexity of Markov decision processes. *Mathematics of Operations Research*, *12*(3), 441–450.

Pineau, J., Gordon, G. J., & Thrun, S. (2006). Anytime point-based approximations for large POMDPs. *Journal of Artificial Intelligence Research*, *27*, 335–380.

Prajapati, A. K., Roy, B. K., & Prasad, R. (2018). A state of the art review of integrated vehicle health management systems. In *IEEE International Conference for Convergence in Technology*.

Ragi, S., & Chong, E. K. P. (2014). UAV path planning in a dynamic environment via partially observable Markov decision process. *IEEE Transactions on Aerospace and Electronic Systems*, *49*(4), 2397–2412.

Rasmussen, R. D. (2008). GN&C fault protection fundamentals. In *AAS Guidance and Control Conference*.

Reason, J. (1990). *Human error*. Cambridge University Press.

Rebeschini, P., & Van Handel, R. (2015). Can local particle filters beat the curse of dimensionality? *Annals of Applied Probability*, *25*(5), 2809–2866.

Rigamonti, M., Baraldi, P., Zio, E., Astigarraga, D., & Galarza, A. (2016). Particle filter-based prognostics for an electrolytic capacitor working in variable operating conditions. *IEEE Transactions on Power Electronics*, *31*(2), 1567–1575.

Ross, S., Pineau, J., & Paquet, S. (2008). Online planning algorithms for POMDPs. *Journal of Artificial Intelligence Research*, *32*, 663–704.

Roy, N., Gordon, G. J., & Thrun, S. (2005). Finding approximate POMDP solutions through belief compression. *Journal of Artificial Intelligence Research*, *23*, 1–40.

Roychoudhury, I., & Daigle, M. J. (2011). An integrated model-based diagnostic and prognostic framework. In *International Workshop on the Principles of Diagnosis*.

Schwabacher, M., Samuels, J., & Brownston, L. (2002). NASA integrated vehicle health management technology experiment for X-37. In *SPIE AeroSense Conference*.

Silver, D., & Veness, J. (2010). Monte-Carlo planning in large POMDPs. In *Advances In Neural Information Processing Systems*.

Snyder, C., Bengtsson, T., Bickel, P., & Anderson, J. (2008). Obstacles to high-dimensional particle filtering. *Monthly Weather Review*, *136*(12), 4629–4640.

Spaan, M. T., Gonçalves, N., & Sequeira, J. (2010). Multirobot coordination by auctioning POMDPs. In *IEEE International Conference on Robotics and Automation*.

Spaan, M. T., Veiga, T. S., & Lima, P. U. (2015). Decision-theoretic planning under uncertainty with information rewards for active cooperative perception. *Autonomous Agents and Multi-Agent Systems*, *29*, 1157–1185.

Tang, L., Hettler, E., Zhang, B., & Decastro, J. (2011). A testbed for real-time autonomous vehicle PHM and contingency management applications. In *Annual Conference of the Prognostics and Health Management Society*.

Tang, L., Roemer, M., Bharadwaj, S., & Belcastro, C. (2008). An integrated health assessment and fault contingency. In *AIAA GN&C Conference*.

Ure, N. K., Chowdhary, G., How, J. P., Vavrina, M. A., & Vian, J. (2013). Health aware planning under uncertainty for UAV missions with heterogeneous teams. In *European Control Conference*.

Valasek, J. (2012). *Advances in intelligent and autonomous aerospace systems*. AIAA.

Virkler, D. A., Hillberry, B., & Goel, P. K. (1979). The statistical nature of fatigue crack propagation. *Journal of Engineering Materials and Technology*, *101*(2), 148–153.

Wang, P., Youn, B. D., & Hu, C. (2012). A generic probabilistic framework for structural health prognostics and uncertainty management. *Mechanical Systems and Signal Processing*, *28*, 622–637.

Weld, D., Anderson, C., & Smith, D. (1998). Extending Graphplan to handle uncertainty & sensing actions. *AAAI/IAAI Conference*.

Williams, B. C., & Pandurang, P. (1996). A model-based approach to reactive self-configuring systems. In *AAAI National Conference on Artificial Intelligence*.

Ye, N., Somani, A., Hsu, D., & Lee, W. S. (2017). DESPOT: online POMDP planning with regularization. *Journal of Artificial Intelligence Research*, *58*, 231–266.