An Approach to Prognostic Decision Making in the Aerospace Domain

Edward Balaban[1], Juan J. Alonso[2], and Kai F. Goebel[3]

[1] *NASA Ames Research Center, Moffett Field, CA, 94035, USA*
edward.balaban@nasa.gov

[2] *Stanford University, Stanford, CA, 94305, USA*
jjalonso@stanford.edu

[3] *NASA Ames Research Center, Moffett Field, CA, 94035, USA*
kai.goebel@nasa.gov

ABSTRACT

The field of Prognostic Health Management (PHM) has been undergoing rapid growth in recent years, with development of increasingly sophisticated techniques for diagnosing faults in system components and estimating fault progression trajectories. Research efforts on how to utilize prognostic health information (e.g. for extending the remaining useful life of the system, increasing safety, or maximizing operational effectiveness) are mostly in their early stages, however. This process of using prognostic information to determine a system's actions or its configuration is beginning to be referred to as Prognostic Decision Making (PDM). In this paper we, first, propose a formulation of the PDM problem with the attributes of the aerospace domain in mind, outline some of the key requirements on PDM methods, and explore techniques that can be used as a foundation of PDM development. The problem of Pareto set viability, i.e. satisfaction of performance goals set for objective functions, is discussed next, followed by ideas for possible solutions. The ideas, termed Dynamic Constraint Redesign (DCR), have roots in the fields of Multidisciplinary Design Optimization and Game Theory. Prototype PDM and DCR algorithms are also described and results of their testing are presented.

1. INTRODUCTION

As aerospace vehicles become more complex and their missions more demanding, it is becoming increasingly challenging for even the most experienced pilots, controllers, and maintenance personnel to analyze changes in vehicle behavior that can indicate a fault and accurately predict the short- and long-term effects a fault can produce. Therefore, some of the latest vehicle designs begin to incorporate automated fault diagnostic and prognostic methods that can assist with these tasks. The research into how to utilize prognostics-enabled health information in making autonomous or semi-autonomous decisions on system reconfiguration or mission replanning is still in its early stages, however.

There are other fields (e.g. operations research, medicine, financial analysis, and weather forecasting) where computer-assisted Prognostic Decision Making (PDM) could or already plays a role - even if the terminology used for it is different. While the fundamentals of PDM methods for these fields are likely to be similar, we believe that there are important reasons to examine how such methods should be developed and used specifically in the context of aerospace.

First, we believe that PDM development needs to be informed by the unique set of aerospace domain characteristics, where operating environment is often harsh and dynamic, systems are highly complex, and an incorrect decision can lead to loss of life. Conversely, it would be beneficial to inform vehicle design by the needs and capabilities of PDM algorithms. This includes computing requirements, sensor suit selection, component redundancy considerations, operating procedures, and communication architectures. A capable (and appropriately verified and validated) PDM system can expand both design and operating options for an aerospace vehicle in much the same way as a new composite material for its structure or a new type of fuel for its propulsion system.

We foresee a number of use cases for PDM in aerospace applications, with some possibilities listed below:

- Maintenance and supply chain management
- Safety assurance for manned aircraft and spacecraft

- Mission effectiveness maximization for unmanned vehicles

In this paper we discuss the general properties that problems of interest to PDM researchers have and consider how methods from the fields of mathematical optimization, multidisciplinary design optimization, and game theory can be used in development of PDM systems for aerospace. The paper is organized around the following objectives:

- Provide some motivating examples for considering PDM in the context of aerospace (Section 3)
- Outline requirements that characteristics of aerospace domain impose on PDM methods (Section 4)
- Discuss some of the approaches to building PDM systems satisfying such requirements (Section 6)
- Define the class of problems where changes in system constraints are required in order to arrive at satisfactory solutions and suggest potential ways of doing that (Section 8)
- Describe prototype algorithms for generating sets of PDM solutions and adjusting system constraints (Section 7 and Section 9, respectively)
- Demonstrate the algorithms on example scenarios involving a planetary rover (Section 11).

Additionally, the next section (Section 2) provides the formal definitions used in this work, Section 5 contains a review of related prior efforts, and Section 10 describes the software/hardware testbed used in the experiments. The paper is concluded with a summary of findings and an outline of potential directions for future work.

2. Definitions

2.1. System

A system considered in this work is formulated generally as Partially Observable Markov Decision Process (POMDP). It is defined as a tuple $\{S, A, O, b_0, T, \omega, Z, R\}$, with the components explained below:

S	A finite set of partially observable states
A	A finite set of possible actions
O	A finite set of observations
b_0	An initial set of beliefs
$T : S \times A \rightarrow P(S)$	A state transition function, for each state and action giving a probability distribution over states $p(s'\|s, a) = T(s, a, s')$.
$\Omega : A \times S \rightarrow P(Z)$	An observation probability function (sensor model)
$R : S \times A \rightarrow \Re$	A reward function

2.2. Prognostics

Prognostics is defined as the information on projected change in plant behavior through time, e.g. due to wear or degradation (Figure 1). The curve on the figure is drawn for just one of the plant parameters. HI axis represents a generalized *Health Index*, with $HI = 1$ corresponding to full health (relative to the parameter).

Prognostic estimates are often represented as probability distributions:

$\Phi_{t_d}(t_p, L(t)) = p(s(t_p)|L(t_d : t_p))$, where

t_d	the time instance for which the function is defined
t_p	the time instance of the prediction
$L(t)$	operating conditions, estimated as a function of time

The following assumptions are made for the above definition:

- A prognostic function is defined for a specific instance in time, given the information up to that moment
- The function is defined for a specific component or system (can be represented by plant model M)
- Prognosis depends on estimation of future operating conditions
- Uncertainty in plant model outputs, observations, and $L(t)$ is possible

Additionally, constraints on the system are defined at a particular time t as **inequality constraints** and **equality constraints**, respectively:

$G_t(\pi) = \{g_1(\pi), g_2(\pi), ...g_N(\pi)\}, g_i(\pi) \geq 0, i = 1, 2, ...N$
$H_t(\pi) = \{h_1(\pi), h_2(\pi), ...h_M(\pi)\}, h_i(\pi) = 0, i = 1, 2, ...M$

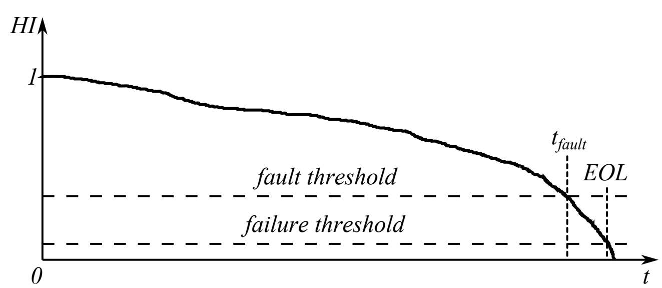

Figure 1. Degradation progression

End of Life (EOL) is then defined as:

$t_{EOL} = \min t : ((\exists g_i \in G_i : g_i < 0) \vee (\exists h_i \in H_i : h_i \neq 0))$

Violating one of the constraints corresponds to crossing of the **failure threshold**. Crossing the **fault threshold** indicates that the parameter is now outside of the nominal performance bounds, although no failure constraints have been violated yet. In practice, the prognostic process is often started after a fault threshold has been crossed. A fault is considered to be **significant** if it affects the output of one or more of the objective functions (defined in the next section).

Finally, the **Remaining Useful Life (RUL)** is defined as

$$RUL = t_{EOL} - t_d$$

2.3. Decision Making

Decision-making is then defined as the process of determining an ordered set of actions (a **policy**) that transitions the system into a desired final state:

$$\pi = \{a_1, a_2, ..., a_K | b_s, s_f\},$$

with Π defined as the set of all possible policies

A **satisfactory** or **feasible** policy is defined as

$$\pi = \{a_1, a_2, ..., a_K | b_s, s_f, G, H\}$$

If, additionally, **objective functions** and an **objective vector** are defined

$$f(\vec{\pi}) = \{f_1(\pi), f_2(\pi), ...f_L(\pi)\}$$

then the **optimal policy** can be defined as:

$$\pi_0 := \pi_s : min\vec{f}(\pi_s),$$

where every objective function is reaching its minimum (best) value. Note that a general assumption of multiple objective functions is made.

Finding this strictly optimal (often called **ideal** or **utopian**) policy in practice is usually not possible. Therefore the concept of a compromise policy that achieves good results for the entire objective vector, while possibly not maximizing any particular objective function, is utilized. This concept, known as Pareto optimality, is used widely in economics, operations research, and engineering.

A **Pareto optimal policy** is defined as a policy that is not dominated by any other policy in Π. A vector $\vec{\alpha} = \{\alpha_1, \alpha_2, ..., \alpha_k\}$ is defined to dominate vector $\vec{\beta} = \{\beta_1, \beta_2, ..., \beta_k\}$ if and only if it is partially less than $\vec{\beta}$:

$$(\forall i \in [1, 2, ...k], \alpha_i \leq \beta_i) \wedge (\exists j \in [1, 2, ...k] : \alpha_i < \beta_i)$$

Dominance of $\vec{\alpha}$ over $\vec{\beta}$ is conventionally denoted as $\vec{\alpha} \prec \vec{\beta}$.

Policy $\pi^* \in \Pi$ is then Pareto optimal if and only if:

$$(\forall i = 1, 2, ...K, \neg\exists\pi' \in \Pi : \pi' \neq \pi_*, f_i(\pi') \leq f_i(\pi_*)) \\ \wedge(\exists j = 1, 2, ...K : f_j(\pi) < f_j(\pi_*))$$

π^* is rarely unique, so **Pareto set** (also known as **Pareto front** (or **Pareto frontier)**) is defined:

$$\Pi^* := \{\pi \in \Pi | \neg\exists\pi' \in \Pi, \pi' \prec \pi\}$$

A visual representation of a Pareto front for two objective functions is provided on Figure 2. Note, in particular, that a Pareto front should not be assumed to be continuous or convex.

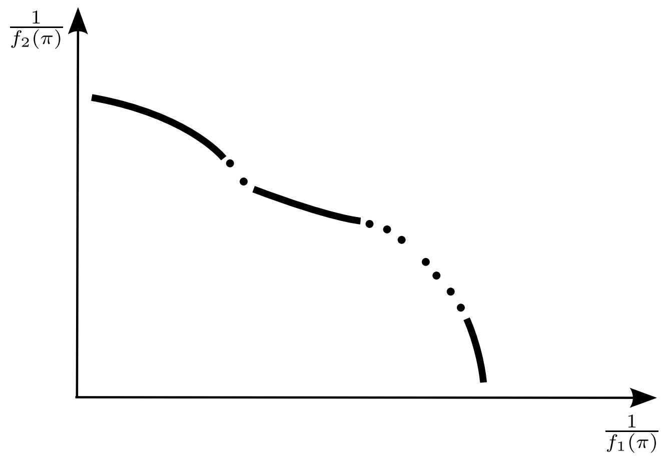

Figure 2. Pareto front

3. MOTIVATING EXAMPLES

Before we propose high-level requirements for an aerospace PDM system, it may be helpful to consider a few motivating examples. They are chosen to illustrate the use cases listed in the Introduction. While only three examples are mentioned here, the aerospace field certainly has no shortage of them.

3.1. A surveying UAV

Our first example is an electrically-powered surveying UAV, such as the SWIFT (Denney & Pai, 2012). The SWIFT is currently in development at NASA Ames Research Center. This example is meant to illustrate the first and the third use cases, that is where PDM could be an integral part of maintenance and logistics operations, as well used for contingency management if degradation of one of the components crosses into the fault region during the mission.

Description

- The UAV performs surveying missions over a defined area (e.g., earthquake fault zone mapping, pipeline monitoring, or air sampling)
- Maintenance for degrading or damaged components needs to be scheduled and replacement parts need to be ordered. Each of the objectives listed below has an importance value associated with it that can change from mission to mission or even within the same mission (if, for instance, an in-flight fault or failure occur).

Objectives

- Maximize the number of measurements or area coverage per mission
- Maximize vehicle availability for missions
- Maximize safety
- Minimize operational costs

Constraints

- Airspace restrictions
- Battery capacity
- Component operating limits
- Return to point of launch (desirable)

3.2. United Airlines Flight 232

The second use case (safety assurance for manned vehicles) is illustrated with the example of United Airlines Flight 232 from Denver to Chicago in 1989 (NTSB, 1989):

Description

- A fan disk in one of the three engines of the DC-10 aircraft failed and disintegrated
- Fan disk shrapnel disabled hydraulic controls
- The crew resorted to using differential thrust on the remaining two engines to steer the aircraft to an emergency landing

Objectives
Minimize injuries and fatalities

Constraints

- Component capabilities and safety margins
- Location and configuration of potential emergency landing sites
- Availability of emergency services at the sites

3.3. Hayabusa (MUSES-C) spacecraft

The example of JAXA's Hayabusa spacecraft (Kawaguchi, Uesugi, & Fujiwara, 2003) illustrates the third use case and is interesting for a number of reasons. While it became the first mission to return samples from an asteroid (Itokawa), it was, however, primarily a technology development mission, with engineering goals assigned point values pre-launch (table 1 [1]). Due to long communication delays during certain phases of the mission, autonomous operation was utilized extensively. Several problems jeopardized mission objectives, however, and required numerous changes to the mission plan and the configuration of the spacecraft.

Description

Table 1. Pre-launch mission goals for Hayabusa

Pre-launch mission goals	Points
Operation of ion engines	50
Operation of ion engines for more than 1000 hours	100
Earth gravity assist with ion engines	150
Rendezvous with Itokawa using autonomous navigation	200
Scientific observations of Itokawa	250
Touch-down and sample collection	275
Capsule recovered	400
Samples obtained for analysis	500

- A large solar flare damaged solar cells en route to the asteroid
- Reduction in electrical power negatively affected the efficiency of the ion engines
- Two reaction wheels (X and Y) failed
- Release of MINERVA mini-probe failed

Objectives
Maximize engineering and scientific payoff

Constraints

- Component capabilities and safety margins
- Orbital mechanics
- On-board propellant amount

4. Problem Class of Interest and Requirements

The examples in the previous section (and others like them) allow us to outline the attributes of the general problem class where, we believe, integration of PDM into the system could have a significant positive impact:

Attributes of the problem class of interest

- The system is complex, consisting of multiple distinct components
- The operating environment is complex and dynamic
- The system may experience degradation processes, due to either external or internal factors, that lead to faults that can be considered significant. Fault magnitudes and secondary effects may evolve over time.
- In case of a fault (or faults), decision on mitigation actions required in a limited amount of time

Requirements

[1] reproduced from http://www.isas.jaxa.jp/e/enterp/missions/hayabusa/today.shtml

The following high-level PDM system requirements can then be derived from the above list:

1. **Should be general and adaptable**
   It may not be possible to define even partial solutions *a priory* for specific combinations of system state, environmental conditions, constraints, and objectives.
2. **Should utilize prognostic information, if available**
   While offering the benefits of an insight into a future system state, incorporation of prognostic capability may also result in substantial increase in computational complexity. In practice, obtaining prognostic information could require execution of a computationally-expensive simulation for each potential solution.
3. **Shall accommodate uncertainty and inconsistency in input data**
   Input data available in aerospace applications often suffers from noise, drop-outs, uncertainty of accuracy, and other issues.
4. **Should not depend on knowing objective function properties**
   Objective functions are not guaranteed to be convex or differentiable, for example, thus 'blackbox' reasoning techniques may need to be utilized.
5. **Shall be time-boundable**
   In most cases a valid solution will be required within a prescribed period of time. In some circumstances the system will also be required to be **interruptable**, i.e. capable of supplying a valid solution even if the decision making process is interrupted before the originally specified time interval has elapsed.
6. **Shall support multi-action policy generation**
   In addition to being able to generate single-action solutions, such as setting controller gain values, the system needs to be able to generate multi-action solution sequences (policies).
7. **Should support system decomposition**
   Ability to account for condition, objectives, and constraints of individual subsystems and components can result in increased solution quality. Carrying out decision-making in a distributed fashion can also be beneficial from the performance point of view.
8. **Should support multiple objectives**
   This requirement is motivated by scenarios where, for instance, failure risk is to be minimized while mission payoff is to be maximized. Also applicable to cases where condition of multiple subsystems or components needs to be taken into account.

A subset of these requirements (*High*-dimensional, *Expensive* (computationally), *Blackbox*) is sometimes referred to in the literature as HEB (Shan & Wang, 2009).

5. PRIOR WORK

Now that our high-level requirements for a PDM system have been outlined and before moving on to describing the approach that we chose to take, we will review some of the prior related efforts. The research efforts described in this section were picked from several different fields where prognostic-style information is used for action determination and we believe them to be representative of the current state of the art.

5.1. Prognostics-enhanced control

Pereira *et al* propose a Model Predictive Control (MPC) approach for actuators that distributes control effort among several redundant units based on prognostic information on their deterioration (Pereira, Galvao, & Yoneyama, 2010). A degradation model of the plant is used that assumes damage accumulation to be proportional to the exerted control effort u and its variation Δu. Bogdanov *et al* (Bogdanov, Chiu, Gokdere, & Vian, 2006) investigate coupling of a prognostic lifetime model for servo motors with a family of LQR controllers. External load disturbance on the servo is assumed to be stochastic.

In (D. W. Brown, Georgoulas, & Bole, 2009) Brown *et al* report on prognostics-enhanced fault-tolerant controller that trades off performance for RUL. The controller is based on MPC principles, with control boundaries for the t_{RUL} corresponding to a particular input u_{RUL} used as soft cost constraints. The work is extended with error analysis and estimation of uncertainty bounds for long-term RUL predictions in (D. W. Brown & Vachtsevanos, 2011). In (Bole, Tang, Goebel, & Vachtsevanos, 2011) Bole *et al* also study the optimal load allocation given prognostic data about fault magnitude growth (including uncertainty bounds on the prediction). *Value at Risk (VaR)*, coming from the field of finance, is used as the key performance metric concept. The case study used in the experiments is an unmanned ground vehicle (UGV) which experiences winding insulation degradation in the drive motors due to thermal stress.

5.2. Post-prognostic decision support and condition-based maintenance

Iyer *et al* use the term *post-prognostic decision support* to describe their framework for generating a Pareto set of possible solutions and for interactive expression of user preferences in the optimization process (Iyer, Goebel, & Bonissone, 2006). The approach is illustrated with a logistics planning example, where mission assets need to be allocated based on the estimated state of health of an asset and projected availability of replacement parts. An exhaustive search technique was used as the optimization method in the experiments, with the intention to replace it with a genetic algorithm in the future.

In (Haddad, Sandborn, & Pecht, 2011b) and (Haddad, Sandborn, & Pecht, 2011a) Haddad *et al* present a prognostics-enabled optimization model for maximization of an offshore wind farm availability. The model is based on Real Options Analysis (ROA) and stochastic dynamic programming. The concept of ROA comes from the field of finance and refers to analysis over either real, tangible assets or opportunities for cost avoidance. The method is illustrated with an example where an optimum subset of turbines to be maintained needs to be found, given information on their degradation, availability requirements, and cost constraints.

5.3. Automated contingency management

The work done by Tang, Edwards, Orchard, and others in Automated Contingency Management (ACM) includes elements of prognostics-enhanced control, but also extends to prognostic mission replanning (Tang et al., 2007; Edwards, Orchard, Tang, Goebel, & Vachtsevanos, 2010; Tang, Hettler, Zhang, & Decastro, 2011). Diagnostic and prognostic algorithms for various types of components were developed and integrated into a prototype decision-making framework for an unmanned ground vehicle (UGV). RUL estimates were used either as a constraint or as an additional element in the cost function of the path-planning algorithm. A *Field D**-style search algorithm was used for receding horizon planning. Methods for estimating and managing process uncertainty were also developed.

6. Policy Generation Approach Selection

Having discussed prior efforts in related areas, we now outline our process of selecting a suitable policy generation approach. At a minimum, all such approaches are expected to be able to produce feasible policies (if they exist). Beyond that, however, the approaches can be divided into two general categories: those that attempt to achieve Pareto optimality and those that do not. The goal of those in the former category is to find solutions on the Pareto frontier or as close to it as possible. The latter often rely on techniques such as decision-trees or contingency planning to generate policies. They are used extensively in many types of applications and often have the advantage of being computationally inexpensive. They also generally lend themselves well to validation and verification.

In this work, however, we chose to formulate the decision-making problem from an optimization point of view - primarily because we believe that this will allow us to take greater advantage of prognostic information. In the rest of the section we briefly comment on the optimization approaches we considered as candidates. More detailed descriptions of modern optimization methods are available, for example, in (Das & Chakrabarti, 2005), (Shan & Wang, 2009), or (Rao, 2009).

Exhaustive search (or **brute-force methods**) are generally easy to implement and are capable of generating exact Pareto sets. Scalability is the main issue with this type of methods, as they quickly become computationally prohibitive. They are, however, useful for verifying performance of other optimization methods on simple problems.

Gradient Descent, Hill Climbing and similar local search methods are not guaranteed to find global optima. Gradient Descent methods also generally require objective functions to be defined and differentiable over the entire search space. **Linear Programming, Constraint Programming, Newton, and Quasi-Newton** methods require knowledge of objective function properties as well.

Dynamic Programming (DP) methods are widely used in policy generation. The main downsides of traditional DP formulations are that for multi-objective problems a single composite objective function needs to be constructed, i.e. a Pareto set is not produced, and that system decomposition can be difficult to accomplish. Some DP-based methods have been developed, however, that attempt to circumvent both of these issues (see (Hussein & Abo-Sinna, 1993; Driessen & Kwok, 1998; Liao, 2002)).

Stochastic methods, such as Simulated Annealing, Quantum Annealing, Metropolis-Hastings, Cross-Entropy, or Probability Collectives generally satisfy the requirements we defined in Section 4. None of them guarantee optimality; they do, on the other hand, posses the *anytime* property (can be interrupted at any time and still return a valid result), can be used with *black box* objective functions, and can accommodate system decomposition.

Genetic algorithms (often classified together with stochastic methods) also satisfy the proposed requirements. In such algorithms a prototype (candidate) solution is described as an individual member of a population. Biologically inspired operators (selection, reproduction, mutation, and others) are used, guided by fitness functions. Genetic algorithms produce a Pareto front approximation in each iteration and thus are also *anytime*.

For this phase of our work we ultimately chose to develop a method based on Probability Collectives for policy generation and a method based on Simulated Annealing for the constraint redesign framework (Section 9). In the future, we also plan investigate policy generation via a genetic algorithm.

7. Policy Generation Algorithm Development

The current policy optimization algorithm, referred to as Probabilistic Policy Generator (PPG), has its roots in the work on *Probability Collectives* (Wolpert, Strauss, & Rajnarayan, 2006). It uses 'look-ahead' sampling to aggregate information about policy options, gradually increasing the probability of choosing the more optimal solutions (although global optimality is not guaranteed). Its input parameters are the following:

A	valid actions set
$\vec{f}(\pi)$	objective functions vector
$\vec{v}$	objectives preferences vector
G	inequality constraint set
H	equality constraint set
l	maximum policy length
N_1	number of utility function calls allocated to the first phase of the algorithm
N_2	number of utility function calls allocated to the second phase of the algorithm
M	number of stages

Execution time is controlled by specifying l, N_1, N_2, and M (further explained below). The algorithm (see Algorithm 1) operates in the following manner:

Initialization (lines 2-5)
A set of policy roots, Π', is initialized with a single member, π'_0, containing a single action to assume the starting state. A policy root is an initial segment of a policy. For instance, $\{a_1, a_2\}$ is a root of $\{a_1, a_2, a_3, a_4, a_5\}$. The probability of π'_0 is set to 1. Finally, first phase utility function calls are allocated per stage (for a total of N_1), with increasing stage numbers corresponding to progressively longer policy roots. The allocation is currently done using a cubic function, with the earlier stages receiving a greater proportion of the quota.

Policy root extension (lines 7-15)
The first phase of the algorithm is executed for M number of stages. In each iteration the policy roots in Π', generated during the preceding stages, are extended and their probability is estimated. In order to extend the roots, sets of possible follow-on actions are determined first. In the example problem described in Section 11, the rover should visit each of its target locations once at the most. Thus, if five locations maximum are to be visited, root $\pi' = \{a_1, a_2\}$ (move to a_1, then to a_2) has $A_{\pi'} = \{a_3, a_4, a_5\}$ as the set of possible follow-on actions. The valid one-action extensions are then $\{a_1, a_2, a_3\}$, $\{a_1, a_2, a_4\}$, and $\{a_1, a_2, a_5\}$. These *offsping* policy roots replace the *parent* root (π') in Π' and split its probability value evenly.

Policy roots probability estimation (lines 17-22)
The probability of each root in updated Π' achieving maximum utility is estimated next. To achieve that, Π' is sampled randomly according to the prior distribution. Each sample root π'_s is extended to the maximum policy length l, with valid completion actions selected from $A_{\pi'_s}$. A sample policy π_s is then evaluated with respect to the objective functions vector $\vec{f}$ and constraint sets H and G. Note that in order to satisfy the constraints, the extended policy may be truncated short of the maximum length. For instance, if $\pi_s = \{a_1, a_2, a_3, a_4, a_5\}$ does not satisfy one or more of system constraints, while $\pi_s = \{a_1, a_2, a_3, a_4\}$ does, then the latter is picked. Utility value u_s is computed for π_s (currently by using the preference vector $\vec{v}$) and posterior probability of the root policy π'_s is adjusted after the sampling process is complete. Normalized Root Mean Squared Error (NRMSE) metric is used to aggregate information on how well π'_s is performing relative to the maximum utility value seen so far.

Monte Carlo simulation on Π' (lines 27-32)
Once the probability distribution over policy roots in Π' is shaped, a Monte Carlo simulation is run for N_2 samples. Policy roots are picked according to the distribution, extended to maximum length satisfying H and G and evaluated with respect to $\vec{f}$.

Solution set filtering (lines 36-38)
Finally, the solution set Π^* is reduced using a variant of the bounded objective method and according to the optimization priority vector $\vec{v}$. The objective functions in $\vec{f}(\pi$ are sorted in the descending order according to the values in $\vec{v}$, $|v| = K$. Π^* is then reduced to $\Pi^*_{f_1}$, where the highest-ranked objective is maximized. $\Pi^*_{f_1}$ is then reduced to $\Pi^*_{f_2}$ and so on, until either $|\Pi^*_{f_k}| = 1$ ($k = 1, 2, ...K$) or $k = K$.

8. DYNAMIC CONSTRAINT REDESIGN

The preceding section of the paper concentrated on selection of promising optimization methods for PDM and methods of incorporating prognostic information into the decision-making process. The outcome of the process is a Pareto set of policies Π^*. There are three "goldilocks" possibilities with regard to the size of Π^*:

1. The size of the set is acceptable, i.e. $1 \leq |\Pi^*| \leq M$, where M is the maximum number of candidate policies that can be practically be downselected by inspection, through a heuristic, or some other method.
2. The size is too large, i.e. $|\Pi^*| \geq M$. In this case the set can be reduced either through interaction with a human expert (as described earlier in (Iyer et al., 2006)) or through an autonomous process that adds/modifies constraints in $G(\pi)$ and $H(\pi)$ and re-runs the optimization until Π^* of a desired size is achieved.
3. No feasible solutions exist, i.e. $|\Pi^*| = 0$. In this case the original constraints in $G(\pi)$ and $H(\pi)$ may need to be relaxed or eliminated.

The second case is an interesting research area that we hope to explore more in the future. In the current work, however, we focused on the third case. In addition to absence of feasible

Algorithm 1 PPG

procedure PPG$(A, \vec{f}(\pi), \vec{v}, l, N_1, N_2, M)$
  $\pi'_0 \leftarrow \{a_0\}$ ▷ null action to assume the initial state
  $\Pi' \leftarrow \{\pi_0\}$ ▷ set of all policy roots
  $p(\pi'_0) = 1$ ▷ assign initial probability
  $N_s \leftarrow allocateUtilityFunctionCalls(N_1)$
  for $stage \leftarrow 1, M$ **do**
    for all π'_i in Π' **do**
      $A_{\pi'_i} \leftarrow getValidActions(\pi'_i)$
      ▷ generate all possible one-action extensions
      $\Pi'_{\pi_i} \leftarrow extendPolicyRoot(\pi'_i, A_{\pi'_i})$
      $\Pi'_{new} \leftarrow \{\Pi'_{new}, \Pi'_{\pi'_i}\}$
      for all π'_j in Π'_{new} **do**
        $p(\pi'_j) \leftarrow p(\pi'_i)/\,|\Pi'_{new}|$
      end for
    end for
    ▷ update $P(\Pi')$
    for $i \leftarrow 1, N_s(stage)$ **do**
      $\pi'_s \leftarrow getRandomSample(\Pi', P(\Pi'))$
      $\pi_s \leftarrow extendPolicy(\pi'_s, l)$
      $\vec{f}(\pi_s) \leftarrow evaluatePolicy(\pi_s, \vec{f}(\pi), H, G)$
      $u_s \leftarrow calculateUtility(\vec{f}(\pi_s), \vec{v})$
      $p(\pi'_s) \leftarrow updateRootProbability(\pi'_s, u_s)$
    end for
  end for
  $\Pi' \leftarrow \{\Pi'_{new}\}$
  $\Pi_{mc} \leftarrow \emptyset$
  ▷ Monte Carlo simulation on Π'
  for $i \leftarrow 1, N_2$ **do**
    $\pi'_{mc} \leftarrow getRandomSample(\Pi', P(\Pi'))$
    $\pi_{mc} \leftarrow extendPolicy(\pi'_s, l)$
    $\vec{f}(\pi_s) \leftarrow evaluatePolicy(\pi_s, f(\vec{\pi}), H, G)$
    $\Pi_{mc} \leftarrow \{\Pi_{mc}, \pi_s\}$
  end for
  $\Pi^* \leftarrow \Pi_{mc}$
  ▷ Filter policy set
  $\vec{f}(\pi)_{sorted} \leftarrow sortDescending(\vec{f}(\pi), \vec{v})$
  n=1
  while $(|\Pi^*| \geq 1) \& (k < K)$ **do**
    for all π in Π^* **do**
      $\Pi^* \leftarrow \{$ all π in $\Pi^* | f_k(\pi)$ is $max\}$
    end for
  end while
end procedure

solutions, however, there could be another reason why Π^* may not be suitable.

8.1. Policy viability

Imagine if in addition to constraints in $G(\pi)$ and $H(\pi)$, constraints (or, rather, performance goals) were also defined for some or all of the elements of $\vec{F}$. Using a UAV as an example, this could mean specifying that the remaining range should be sufficient to reach a suitable landing location.

$$\Gamma_t(\vec{F}) = \{\gamma_1(\vec{F}), \gamma_2(\vec{F}), ...\gamma_N(\vec{F})\}, \gamma_i(\vec{F}) \geq 0, i = 1, 2, ...N$$

Only the inequality constraints are currently considered. Although semantically similar to the concept of feasibility, **viable** solutions satisfy constraints set on the objective functions, as well as the actions within policies:

$$\pi_v = \{a_1, a_2, ..., a_K | b_s, s_f, G, H, \Gamma\}$$

An example of a non-viable Pareto set is illustrated on Figure 3. Viability of a solution implies its feasibility, i.e. in order to be viable, a solution first needs to be feasible. If there are no constraints set on objective functions, then viability is equivalent to feasibility.

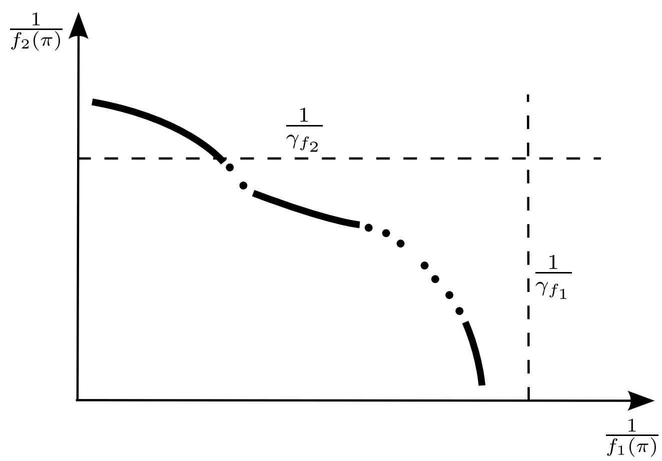

Figure 3. An example of a non-viable Pareto set

If no viable solutions are found during the optimization process and if the objective function constraints (performance goals) are considered to be of high enough importance, then the system constraints may need to be changed or eliminated.

For convenience, in this paper we will refer to the process of changing or eliminating system constraints as **Dynamic Constraint Redesign (DCR)**. In the context of an aerospace vehicle, DCR could mean knowingly damaging a component or a subsystem beyond repair if that means saving the overall vehicle. Only system constraints will be considered for the purpose of this discussion, however one can also envision eliminating or relaxing external constraints, such as airspace restrictions or separation distances.

DCR can also be thought of as redesigning the vehicle ”on the fly”, by changing its performance characteristics outside of the known envelope. Some of the same issues arise as during the original design, e.g. subsystem compatibility assurance, choice of design variables, and their sensitivity analysis. In the last few decades the field of Multidisciplinary Design Optimization (MDO) has been developed to address these and other issues during the initial design of complex systems. We believe that some of the techniques from the MDO community could be beneficial in development of DCR as well.

8.2. Multidisciplinary Design Optimization (MDO)

In this section we briefly review some of the most popular MDO approaches and comment on their applicability to DCR. First, however, it would be helpful to note some key differences between MDO and DCR problems:

- Robust validation and verification of a candidate point design using independent methods, unlike in MDO, may not be possible in PDM/DCR
- Related to the preceding point, risk associated with a DCR decision needs to be quantified and supplied with each potential solution
- Achieving real-time performance will, generally, be far more important to PDM/DCR than to MDO

One of the ways to classify modern MDO algorithms is into the following two broad categories: All-At-Once (AAO) and decomposition. **All-at-Once** algorithms, also referred to in some publications as All-In-One (AIO) or single-level, aim to achieve design decisions through a single global optimization process (Cramer, Dennis, Frank, Shubin, & Lewis, 1993; N. Brown, 2004). While such formulations have some attractive qualities (for instance, each iteration produces a discipline-feasible solution and sensitivity analysis on design variables is usually easy to perform), they also have some significant downsides. A designer using AAO methods is likely to run into scalability issues when applying them to large, complex systems. Also, by aggregating knowledge from the subsystems into a single optimizer, some of the discipline-specific knowledge may be lost.

Decomposition methods distribute the design optimization problem into multiple subproblems, usually along the boundaries of disciplines, subsystems, or individual components (Cramer et al., 1993). Some of the better known methods are bi-level, such as **Collaborative Optimization (CO)**, **Concurrent Subspace Optimization (CSSO)**, or **Bi-Level Integrated System Synthesis (BLISS)**, and multi-level, such as **Analytical Target Cascading (ATC)**.

CO (Braun, Gage, Kroo, & Sobiesky, 1996; Roth & Kroo, 2008; Roth, 2008) uses target values of the design and state variables, specified at the system level, to guide individual discipline optimizations. Communication between disciplines in most CO implementations is limited, which simplifies implementation, but can result in slow convergence.

The **CSSO** method (J. E. Renaud & Gabriele, 1993; Sobieszczanski-Sobieski, Agte, & Sandusky, 1998; Sellar, Batill, & Renaud, 1996; G. Renaud & Shi, 2002) performs discipline-specific optimization using local objective functions, variables, and constraints, while approximating effects on system performance using Global Sensitivity Equations, Response Surfaces, or other types of system models. Similarly, in the system-level optimization models of disciplines are used to approximate their behavior. As performance information is accumulated throughout the process, the models can be updated correspondingly.

In **BLISS** (Sobieszczanski-Sobieski et al., 1998; Sobieszczanski-Sobieski, Emiley, Agte, & Sandusky, 2000) each iteration of the procedure improves the design both on the local (discipline) and system levels. First, a concurrent local optimization is performed using the discipline design variables and keeping the system-level variables constant. Then, a system-level optimization on shared variables is performed. Total derivatives are communicated among the disciplines to help predict the effects of local design choices on the other disciplines.

Analytical Target Cascading (ATC) (Kim, 2001; Kim, Michelena, Papalambros, & Jiang, 2003; Allison, Kokkolaras, Zawislak, & Papalambros, 2005), is primarily intended for problem decomposition by subsystem and components, rather than disciplines. ATC approach is flexible and multi-level, allowing complex system architectures to be represented. Other formal MDO methods can potentially be integrated within an ATC framework (Agte et al., 2009).

Methods founded on the principle of **Lagrangian Duality (LD)** may also be of interest for certain types of applications. Classical LD methods are generally applied to convex problems and accommodate decomposition into smaller sub-problems. In order to handle non-convex problems, Augmented Lagrangian Duality (ALD) theory has been developed (Hestenes, 1969). ALD algorithms, however, lose the decomposition ability. In recent years, several research efforts combined LD and ALD approaches to attain both the ’convexification’ properties of ALD and the decomposition properties of traditional LD (Blouin, Lassiter, Wiecek, & Fadel, 2005; Tosserams, Etman, Papalambros, & Rooda, 2005).

Finally, MDO methods evolved from the field of **Game Theory** offer some promising alternatives for design decomposition architectures. The idea of using game formulations in design problems goes back to the work of Vincent (Vincent, 1983) and Rao and Freiheit (Rao & Freiheit, 1991). Some of the further developments are described in (Lewis & Mistree, 1997), (Marston, 2000), and (Clarich & Pediroda, 2004). Games of different forms have been studied, at least to some extent, for use in MDO applications: cooperative (Pareto), approximately cooperative, non-cooperative (Nash), coalition, and leader/follower. While intuitively a cooperative (Pareto) form game would appear to be the natural choice when setting up an MDO or a DCR problem, the other forms have their place as well. For instance, the leader/follower (also known as Stackelberg or extensive) form can be used to set up a sequential design problem. Non-cooperative (Nash) form could be used in situations when the established communication protocols between subsystems prove to be insufficient for a particular situation or are affected by a system fault. The coalition form can be used to organize system analysis by discipline.

For the first DCR prototype we chose to implement a cooperative game-theoretic protocol (described in the next section), with alternative formulations to be implemented and compared in future work. Similarly to BLISS, the implemented algorithm passes derivatives of local objective functions with respect to shared variables, in order to inform subsystems of effects their choices will have on the other subsystems.

9. DCR ALGORITHM DEVELOPMENT

In the developed game-theoretic DCR algorithm the players (subsystems) cooperate in exploring the (potentially large) option space by taking turns in conducting the search and, when necessary, relaxing some of their constraints. The current formulation of the algorithm tests the concept for a maximum of two players, with extension to larger numbers of players planned for subsequent versions. One constraint per player is currently chosen as target for redesign. Without loss of applicability to equality constraints, in the description that follows we assume that the target subsystem constraints g_{t1} and g_{t2} are inequalities (i.e. $60 - T_{max} > 0$ or $0.5 - v > 0$).

The process (illustrated on Figure 4) starts with one player randomly picked to go first (let us assume that its Subsystem 1). *Subsystem 1* then conducts its iteration of the search, finding its best guess at the optimal policy π^* . Note that the concept of selecting the values of design variables from MDO maps well into the concept of selecting a policy, where in the latter case some of the actions in π could be defined as assigning variable values. The policy needs to satisfy both the system constraints G and H and the constraints placed on the system objective functions (Γ). Also, a maximum of N utility function calls is allowed per iteration. If no viable policy is found, the target constraint g_{t1} is adjusted (becoming, for instance, $62 - T_{max} > 0$). Another search iteration is performed and viability of solutions is evaluated. The process repeats until a maximum number of search tries, M, is reached or a non-empty viable set Π_1^* is found. Π_1^*, empty or otherwise, is then sent over to *Subsystem 2*, along with gradient information on objective function performance (in a non-cooperative formulation only Π_1^*, also known as the Best Reply Correspondence or BRC, would be transmitted). Note that gradient estimates are shared not only for policies in Π_1^*, but also for other policies considered during the search iteration. If there is at least one policy $\pi^* \in \Pi_1^*$ that is found to be viable from the point of view of *Subsystem 2*, then the process is stopped. Otherwise *Subsystem 2* conducts its own search iteration, adjusting g_{t1} as needed, and hands over control of the search to *Subsystem 1* after either M iterations are completed or a non-empty Π_2^* is found. Π_2^* and objective function gradients are then transmitted back to *Subsystem 1*. The process continues until a viable π^* satisfying both subsystems is found.

It is important to take a look at how the objective function for each of the players is designed. In non-cooperative game formulations and some of the other traditional MDO approaches discipline/subsystem objective functions primarily focus on the needs of that particular discipline or subsystem. In this, cooperative, formulation, to help expedite convergence composite objective functions, taking into account the effect a candidate solution may have on global objectives and on the other players, is used. The functions take the following form:

$$f_1(\pi) = w_{1,1} f_g(\pi) + w_{1,2} |\nabla f_{2,l}|_\pi + w_{1,3} f_{1,l}(\pi),$$
$$f_2(\pi) = w_{2,1} f_g(\pi) + w_{2,2} |\nabla f_{1,l}|_\pi + w_{2,3} f_{2,l}(\pi),$$

where f_g is the global objective function (currently a single one), $f_{i,l}$ is the objective function local to the subsystem, i is the subsystem number, and $w_{i,j}$ are the weights used to specify the degree of influence of each of the components of f_i.

Another important feature of the algorithm is that with each iteration the size of the constraint-adjusting step is increased, thus encouraging each player to come up with a solution suitable from the other subsystems' (and global) points of view as quickly as possible.

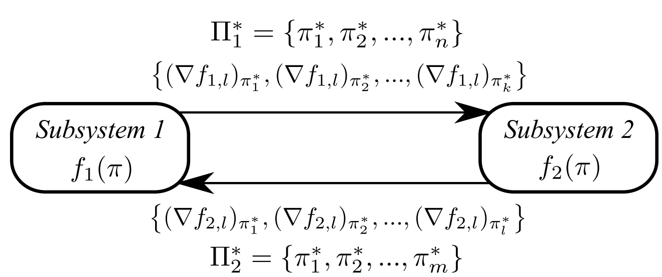

Figure 4. Two-subsystem cooperative game formulation

A variant of Simulated Annealing (SA) is currently utilized for search of the option space by the subsystems, although any suitable optimization algorithm can be used. In this case SA was chosen to take advantage of the gradient information exchanged by the subsystems while avoiding getting 'stuck' at the local minima by performing randomized jumps to other promising locations of the option space. The probability of performing a jump vs. continuing with a local search is influenced by an *annealing schedule*.

10. TEST PLATFORM

The testbed being in the current experiments is the K11 planetary rover prototype and its associated software simulator (Balaban et al., 2011). Another testbed targeted for future validation experiments is the Edge 540 UAV located at NASA Langley (Hogge, Quach, Vazquez, & Hill, 2011). While the algorithmic infrastructure is developed to accommodate the UAV, that part of the work is, otherwise, in its early stages.

10.1. K11 overview

The K11 is a large four-wheeled rover platform (approximately 1.4 m long by 1.1 m wide by 0.63 m tall, weighing roughly 150 kg). Each wheel is driven by an independent

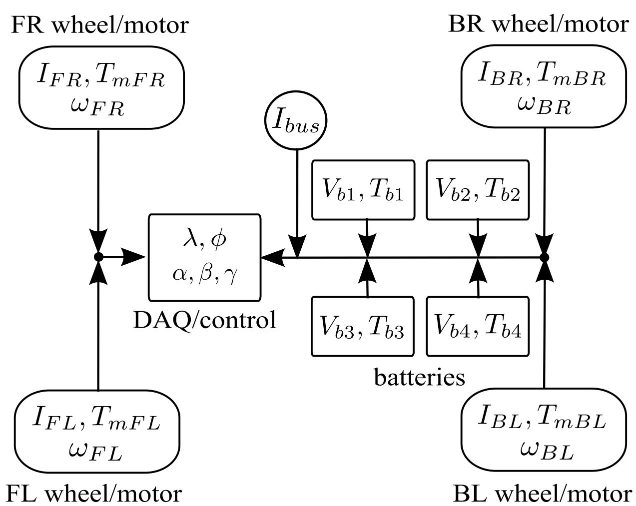

Figure 5. K11 data flow

250 W graphite-brush motor, connected through a bearing and gearhead system, with each motor controlled by a single-axis digital motion controller. Four 14.8 V 3.3 Ah lithium-ion batteries connected in series power the vehicle. The on-board computer runs control and reasoning algorithms, as well as performs data acquisition. Measurements available on-board are shown in Table 2 and on Figure 5.

In the table F, B, L, R refer to front, back, left, and right, respectively. Altitude h is determined using λ, ϕ and a terrain map $\mathcal{M}$.

The software simulator reproduces both nominal and off-nominal behavior of the hardware testbed. The simulator has a dual purpose: (a) to aid in the development of PDM algorithms as a virtual testbed and (b) to provide $\vec{F}$ estimates during the decision-making process.

Table 2. K11 data

measurement	symbol
absolute position (longitude, latitude)	λ, ϕ
wheel angular velocity	$\omega_{FL}, \omega_{FR}, \omega_{BL}, \omega_{BR}$
attitude (yaw, pitch, roll)	α, β, γ
battery temperature	$T_{b1}, T_{b2}, T_{b3}, T_{b4}$
battery voltage	$V_{b1}, V_{b2}, V_{b3}, V_{b4}$
motor temperature	$T_{mFL}, T_{mFR}, T_{mBL}, T_{mBR}$
motor current	$I_{FL}, I_{FR}, I_{BL}, I_{BR}$
power bus current	I_{bus}

10.2. Fault Modes

Table 3 describes the K11 fault modes, implemented either in hardware, simulation, or both. Some of the fault modes, such as sensor faults or motor failure, are injected primarily for testing the diagnostic functionality (i.e. have brief fault-to-failure times), while the others result in a more continuous fault progression behavior and are, therefore, used for validation of prognostic algorithms.

Table 3. K11 fault modes.

fault model	subsystem
battery capacity degradation	Power
parasitic electric load	Power
motor failure	Propulsion
increased motor friction	Propulsion
sensor bias/drift/failure	Sensors

10.3. Diagnostic Functionality

Two diagnostic algorithms are currently in use with the K11 testbed. The first one, QED (Qualitative Event-based Diagnosis), is described in (Daigle & Roychoudhury, 2010). It utilizes a qualitative diagnosis methodology that isolates faults based on the transients they cause in system behavior, manifesting as deviations in residual values (Mosterman & Biswas, 1999). The second, the Hybrid Diagnosis Engine (HyDE) is a diagnosis algorithm that uses candidate generation and consistency checking to diagnose discrete faults in stochastic hybrid systems (Narasimhan, 2007). 'Hybrid' in this case refers to combined discrete and continuous models used by the algorithm to analyze input data and deduce the evolution of the state of the system over time, including changes indicative of faults.

10.4. Prognostic Functionality

Once a fault is detected and diagnosed, a prognostic algorithm appropriate to the type of the fault is invoked. For battery capacity deterioration, as well as charge estimation, an algorithm based on the Particle Filter framework is used (Saha & Goebel, 2009) and (Saha et al., 2011). Prognostic estimation of temperature build-up inside the electric motors - which can lead to winding insulation deterioration and eventual failure - is done using a Gaussian Process Regression algorithm (Balaban et al., 2011). Finally, work is in progress to implement prognostics for electronic components of the motor drive units (such as capacitors and power transistors) using Kalman Filter and Extended Kalman Filter approaches (Celaya, Saxena, & Saha, 2011).

11. Experiments

The following section describes the scenarios used for testing the policy optimization algorithm, PPG, and the constraint redesign algorithm, DCR and the results of the tests. Both of the algorithms have been tested in simulation at this time.

11.1. Policy optimization

11.1.1. Problem formulation

Given:

$\vec{g}(\pi) = \{g_h(\pi), g_e(\pi)\}$	Inequality constraints on available energy and health
$\vec{f}(\pi) = \{f_r(\pi), f_h(\pi), f_e(\pi)\}$	Objective functions on cumulative reward, health, and energy
$\vec{v} = \{v_r, v_h, v_e\}, (v_r, v_h, v_e \in [0,1])$	Optimization preferences vector
$N = \{n_1, n_2, ..., n_L\}$	Nodes (locations) to be visited
$\pi = \{a_1, a_2, ..., a_M\}$	Policy is defined as a set of actions
$a := \{n_i, n_j\}, i \in [1, 2, ...L - 1], j \in [1, 2, ...L - 1]$	An action constitutes a move between a pair of nodes (start and finish)
$a_1 = \{n_1, n_1\}$	The first action is a special case - start traversal at node 1
$a_m = \{n_j, n_k\} \mid (a_{m-1} = \{n_i, n_j\}), (i, j, k \in [1, 2, ...L]), (m \in [2, 3, ...M])$	Any action after the first one needs to start on the node where the previous one finished

Find:

Π^*	Pareto set of policy solutions

11.1.2. Test setup

A synthetic terrain map $\mathcal{M}$ was generated (Figure 6) and ten wayponts (nodes) were selected on it. Each node was assigned a reward value (shown in parenthesis). The bar on the right side of the map and the isolines depict the elevation changes.

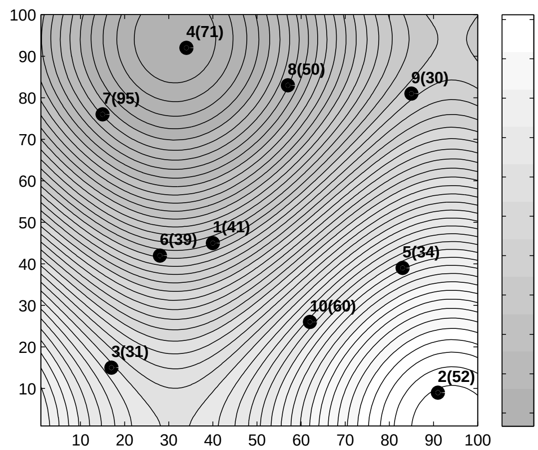

Figure 6. Terrain map with target locations (elevations and distances are in meters)

Test scenarios with increasing numbers of nodes (6-10) were then created. The nodes were selected in such a way so as to make it impossible for the vehicle to visit all of them before either energy depletion or system health deterioration resulted in EOL. PPG was allocated a limited number of utility function calls (UFC) to test performance in resource-constrained conditions. An exhaustive search algorithm (ES), used for verifying PPG results and benchmarking, was not limited in how many times it could invoke the utility function. The metric used for evaluating performance was the cumulative reward for the best path (policy) found by each algorithm. Each scenario was executed 30 times and the mean and standard deviations were computed. All of the code was written in MATLAB (R2010b) and executed on an Intel Core i7 Duo 2.8GHz laptop computer.

11.1.3. Test results

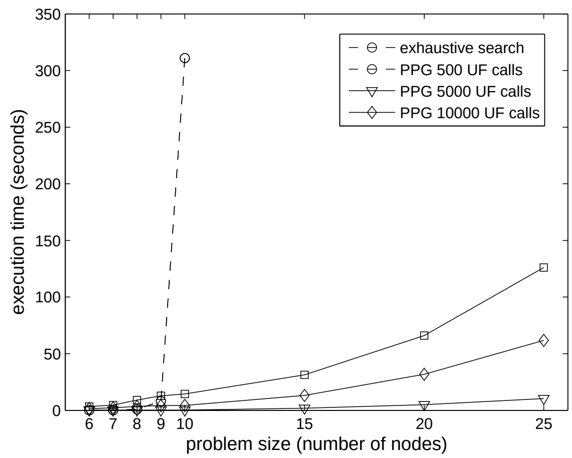

Figure 7. Mean execution times comparison

Table 4 summarizes the cumulative reward results obtained by ES and PPG. The number of utility function calls used by ES is provided for comparison. While not quite achieving scores as high as ES for the larger size problems, PPG still does relatively well, particularly given that in those scenarios it uses a small fraction of UFC used by ES (PPG performance improves, as expected, if more UFC are permitted).

Execution time for each of the algorithms was also recorded for all of the scenarios, with the data summarized in Table 5. It can be observed that execution times for ES start growing exponentially with problem size and using this approach becomes impractical for problems containing more than 10 nodes. While not having the ability to validate the cumulative reward performance on problems larger than that (in reasonable time), we still tested PPG with scenarios containing 15, 20, and 25 nodes. The averaged execution times are presented on Figure 7 and lead us to believe that the approach adopted for PPG remains practical for real-time applications even for policies with large numbers of actions (at least up to 25). The question of how to evaluate the quality of generated policies in large-size problems is something we hope to investigate in subsequent work.

11.2. Dynamic Constraint Redesign

To test the DCR algorithm, a scenario was used where one of the rover motors (FL) has experienced an *Increased Motor Friction* fault. This results in increased current consumption by the motor and, consequently, a higher rate of heat build-up both in it and the batteries supplying the current. For the purposes of this scenario the batteries are viewed as a single unit, with its temperature denoted by T_b. Temperature of the affected motor is denoted as T_m. Even given the fault, the rover is still required to travel a certain distance in a given amount of time in order to reach a point favorable for battery recharging and communication with controllers. To accomplish that, the rover needs to alternate periods of driving with periods of stationary, in order to not exceed the maximum temperature limits for both the battery and the motor. The two components belong to Power (Po) and Propulsion (Pr) subsystems, respectively. As the components heat up and cool at different rates, a suitable schedule for driving and cooling down periods (policy) needs to be negotiated between the subsystems. As no viable policies may exist that satisfy both the minimum distance and the maximum time constraints, the two subsystems may need to negotiate increases in their operating temperature limits. It is in the interest of each subsystem to keep its limit as low as possible, in order to reduce the risk of failure. The rover, as a whole, is also interested in keeping the risk of component failure as low as possible, while still achieving the destination in the time alloted.

11.2.1. Problem formulation

Given:

$v_c = 0.3m/s$	The minimum velocity the rover can maintain without stalling, given the fault. Also assumed to be best (cruise) velocity in terms of energy efficiency
$T_{b,init} = 40°C$	initial operating temperature of the battery
$T_{m,init} = 35°C$	initial operating temperature of the motor
$T_{b,max_0} = 60°C$	initial operating temperature limit for the battery
$T_{m,max_0} = 60°C$	initial operating temperature limit for the motor
$T_a = 30°C$	ambient temperature
$I_s = 5A$	peak current drawn by the affected motor in order to reach v_c from full stop (start current)
$I_c = 2A$	current drawn by the affected motor at v_c (cruise current)
$d_{min} = 500m$	the minimum traverse distance
$t_{max} = 3600s$	the maximum time to reach the destination
$t_s = 2s$	time needed to achieve cruise velocity from a complete stop

Table 4. Maximum cumulative reward values obtained by ES and PPG algorithms (in points)

nodes	ES UFC	ES result	500 UFC PPG mean (σ)	5000 UFC PPG mean (σ)	10000 UFC PPG mean (σ)
6	720	237	235.60 (05.33)	237.00 (00.00)	237.00 (00.00)
7	5040	311	295.80 (11.29)	305.77 (09.11)	305.77 (09.11)
8	40320	343	329.93 (08.51)	342.57 (02.37)	340.83 (04.93)
9	362880	373	326.27 (11.31)	345.47 (16.50)	348.87 (16.76)
10	3628800	403	347.47 (22.93)	382.73 (18.04)	388.97 (16.47)

Table 5. ES and PPG execution time (in seconds)

nodes	ES UFC	ES mean (σ)	500 UFC PPG mean (σ)	5000 UFC PPG mean (σ)	10000 UFC PPG mean (σ)
6	720	0.0192 (0.0003)	0.1756 (0.0049)	1.5961 (0.0183)	3.3543 (0.2253)
7	5040	0.1154 (0.0020)	0.2079 (0.0083)	2.2419 (0.1778)	4.5870 (0.4473)
8	40320	0.8385 (0.0105)	0.2212 (0.0114)	3.5883 (0.2132)	9.0177 (0.8424)
9	362880	8.0367 (0.0448)	0.2350 (0.0072)	4.1321 (0.1315)	12.7910 (0.3515)
10	3628800	310.9904 (3.9258)	0.2412 (0.0060)	4.4041 (0.2654)	14.4285 (0.7322)

A notional current profile for the damaged motor is shown on Figure 8. For simplicity, current draw by the three healthy motors was assumed to be constant throughout the motion at 1A each. It is also assumed that prognostic information on battery and motor EOL is provided.

Find:

t_d	drive period duration
t_c	cooldown period duration
T_{b,max_f}	final operating temperature limit for the battery
T_{m,max_f}	final operating temperature limit for the motor

11.2.2. Test setup

Each subsystem was given a maximum of $M = 3$ search iterations before it had to relinquish control of the process. t_d and t_c could be picked from intervals between 10 and $100s$, in $10s$ intervals. A simplified version of the simulator, tracking only the distance traveled and the temperature state of the affected motor and the battery was used as the utility function, in order to speed up execution. The following general thermal state equation was used in the simulator:

$$dT = \frac{1}{C_t}(RI^2 + h(Ta - T))dt,$$

where T is the component temperature, C_t is the thermal inertia coefficient, R is the resistance, I is the current, h is the heat transfer coefficient, and t_a is the ambient temperature. Model parameters used in the experiments are provided in Table 7.

Table 7. Model parameters

parameter	motor	battery	units
C_t	11	25	$\frac{J}{K}$
R	0.5	1.0	Ohm
h	0.03	0.08	$\frac{W}{K}$

Prognostic information was supplied in a differential form as the probability of reaching EOL:

$$dp_{EOL} = \frac{a}{10^{10}}T^3 dt,$$

where $a = 1.5\frac{1}{K^3}$ for the battery and $a = 1.3\frac{1}{K^3}$ for the motor.

System probability of EOL was calculated as a weighted sum of the two component EOL probabilities: $p_{system} = 0.8p_b + 0.2p_m$. In this case the battery failure was considered to be a greater risk than a motor failure, as in the latter case the possibility of achieving the objective remained by using the remaining three motors. Minimization of the risk of premature failure was included in both the local and the global components of the subsystem objective functions.

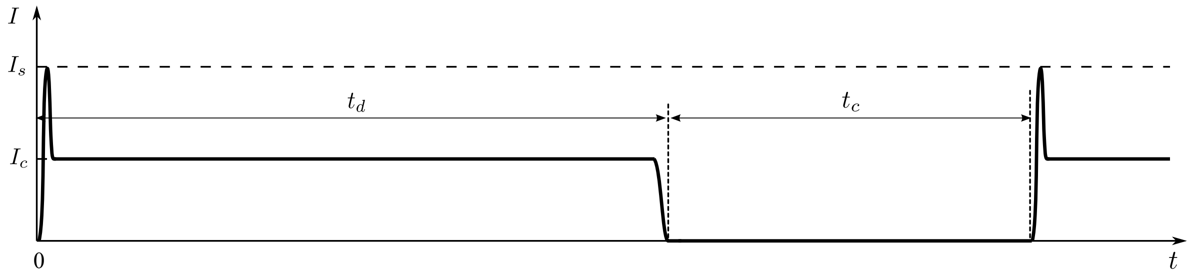

Figure 8. Current profile for the damaged motor

Table 6. DCR iterations in the example run

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
active subsystem	Po	Po	Po	Pr	Pr	Pr	Po	Po	Po	Pr	Pr	Pr	Po	Pr	Pr
$T_d(s)$	50	50	60	30	30	40	70	50	80	40	50	50	80	70	90
$T_c(s)$	80	80	90	90	90	100	100	60	80	90	100	80	80	90	100

11.2.3. Test results

The output from one of the runs of the algorithm is presented on Figure 9 and in Table 6. The top subplot of Figure 9 shows the evolution of temperature constraints for the two subsystems throughout the negotiation process. The middle subplot shows the maximum distances achievable from each of the subsystems' point of view. The process ends when both of the subsystems are predicted to be capable of achieving d_{min}, albeit with a higher risk of failure while doing so. The bottom subplot shows the estimated risk of system failure for each iteration of the algorithm. Table 6 shows which subsystem had the control of the process during each iteration and what policy $\pi^* = \{t_d, t_c\}$ it proposed as the best solution. In the example run given here, the final temperature limit for the motor was found to be at approximately $65.3°C$ and the one for the motor at approximately $75.5°C$.

12. Summary and Future Work

In this paper we outlined our approach to developing prognostic decision making methods for aerospace applications. First definitions for prognostic decision making and related concepts were suggested, then a few motivating examples, highlighting potential use cases for PDM, were described. The examples also helped to illustrate the general attributes of the problem type we hope to address: (1) complex, multi-component systems; (2) dynamic operating environments; (3) degradation/fault modes that evolve in their characteristics over time and have the potential of substantially affect system performance; (4) decisions on mitigation measures required in a finite amount of time. From there we derived our set of high-level requirements for aerospace PDM systems. With these requirements in mind, we reviewed the related prior efforts from the areas of prognostics-enabled control, post-prognostic decision support, condition-based maintenance, and automated contingency management. We then explained our process of selecting suitable policy generation techniques and presented a prototype algorithm that uses probabilistic methods and prognostic information in generating action policies. The algorithm, PPG, was tested against an exhaustive search algorithm on scenarios involving a planetary rover. We also considered the problem when no feasible policies are found or when feasible policies in the generated Pareto set are not sufficient (viable) for attaining performance objectives, given the current constraints. We proposed that this problem has certain common characteristics with problems in the field of Multidisciplinary Design Optimization and reviewed some of the modern MDO approaches for applicability. One of the approaches is based on game-theoretic principles and served as a foundation for the second algorithm presented, DCR. The algorithm sets up a negotiating framework for subsystems to adjust their operating constraints, if that is necessary for achievement of the overall system objectives. DCR was demonstrated on a problem involving two rover subsystems, power and propulsion.

While it is not possible to cover all of the topics discussed in sufficient detail in one paper, we hope that it provides a good foundation for future efforts. The work done so far also gave us a better appreciation for the challenges ahead. One of them is developing more efficient multi-objective optimization algorithms - given the high computational cost of a utility function (simulation) call for a typical application. We plan to continue our development of probabilistic optimization methods and further investigate applicability of evolutionary algorithms. Use of multi-fidelity models and response surfaces for utility simulation will be researched as well.

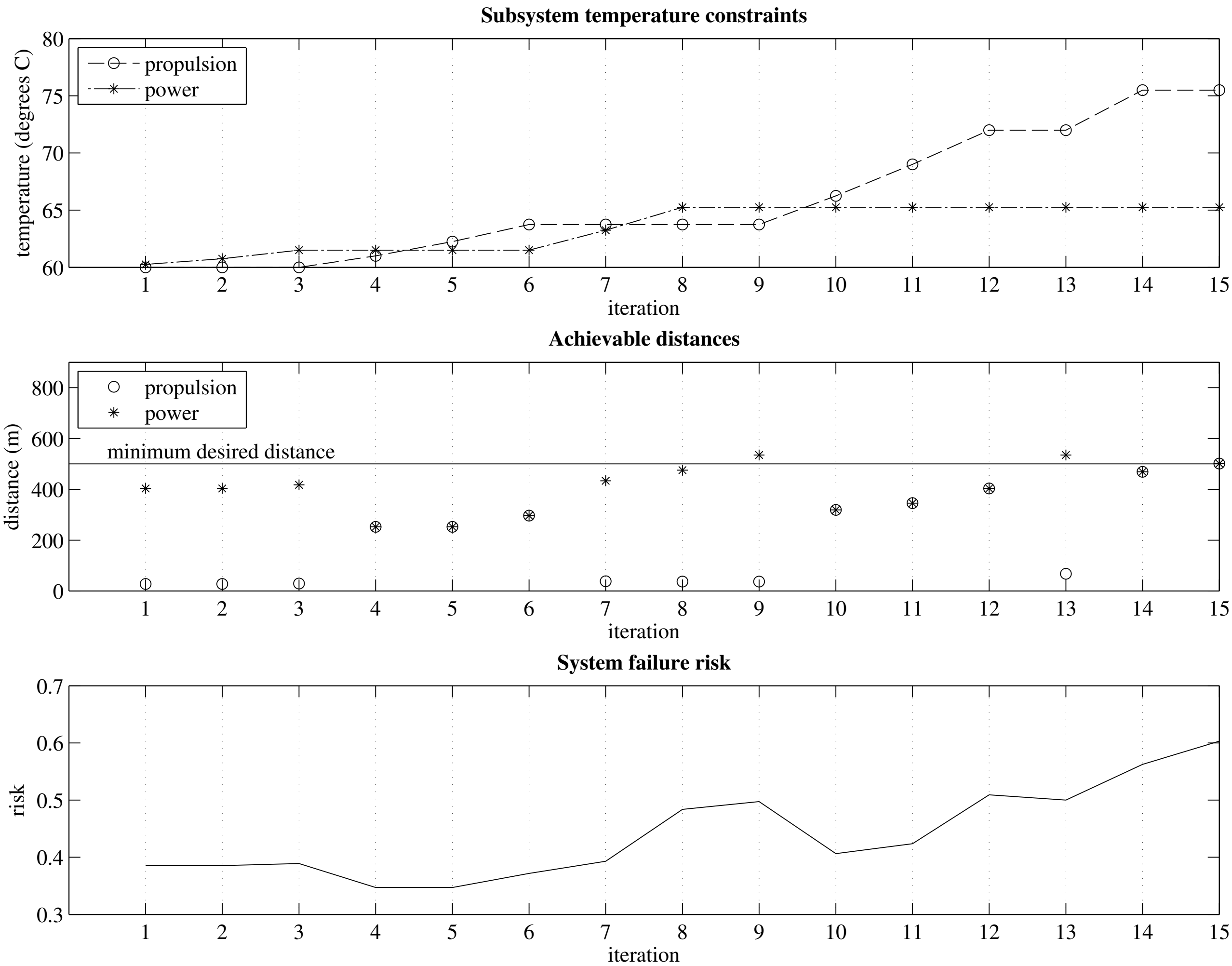

Figure 9. DCR output example

For the problem of DCR, we plan to concentrate on the following three goals: (1) extend the current, cooperative game DCR algorithm to greater numbers of players/subsystems; (2) investigate other formulations, possibly based on ideas in CO, CSSO, and BLISS; (3) develop methods for selection of those constraints that offer the most system benefit if changed (approaches based on Lagrangian Duality appear promising for this purpose). We also hope that decomposition formulations researched for DCR will also prove helpful for the prognostic policy generation work. Finally, identifying and, if necessary, developing suitable performance metrics will become more important as the complexity of both the test scenarios and the algorithms increases.

13. Acknowledgment

The authors would like to thank their colleagues at NASA Ames Research Center, Stanford University, and Los Alamos National Laboratory for stimulating discussions and assistance with this work. Funding from NASA System-Wide Safety Assurance Technologies (SSAT) project (Aviation Safety Program) is also gratefully acknowledged.

References

Agte, J. S., Weck, O., Sobieszczanski-Sobieski, J., Arendsen, P., Morris, A., & Spieck, M. (2009, April). MDO: Assessment and Direction for Advancement - an Opinion of One International Group. *Structural and Multidisciplinary Optimization*, *40*(1-6), 17–33. doi: 10.1007/s00158-009-0381-5

Allison, J., Kokkolaras, M., Zawislak, M., & Papalambros, P. (2005). On the use of analytical target cascading and collaborative optimization for complex system design. In *6th world congress on structural and multidisciplinary optimization* (pp. 1–10). Rio de Janeiro, Brazil.

Balaban, E., Narasimhan, S., Daigle, M., Celaya, J., Roychoudhury, I., & Saha, B. (2011). A Mobile Robot Testbed for Prognostics-Enabled Autonomous Decision Making. In *Annual conference of the prognostics and health management society* (pp. 1–16). Montreal, Canada.

Blouin, V. Y., Lassiter, J. B., Wiecek, M. M., & Fadel, G. M. (2005). Augmented Lagrangian Coordination for Decomposed Design Problems. In *6th world congress on structural and multidisciplinary optimization* (pp. 1–10). Rio de Janeiro, Brazil.

Bogdanov, A., Chiu, S., Gokdere, L. U., & Vian, J. (2006). Stochastic Optimal Control of a Servo Motor with a Lifetime Constraint. In *Proceedings of the 45th ieee conference on decision and control* (pp. 4182–4187). doi: 10.1109/CDC.2006.377205

Bole, B., Tang, L., Goebel, K., & Vachtsevanos, G. (2011). Adaptive Load-Allocation for Prognosis-Based Risk Management. In *Annual conference of the prognostics and health management society* (pp. 1–10).

Braun, R., Gage, P., Kroo, I., & Sobiesky, I. (1996). *Implementation and Performance Issues in Collaborative Optimization* (Tech. Rep.). NASA Langley Research Center.

Brown, D. W., Georgoulas, G., & Bole, B. (2009). Prognostics Enhanced Reconfigurable Control of Electro-Mechanical Actuators. In *Annual conference of the prognostics and health management society* (pp. 1–17). Denver, CO.

Brown, D. W., & Vachtsevanos, G. J. (2011). A Prognostic Health Management Based Framework for Fault-Tolerant Control. In *Annual conference of the prognostics and health management society.* Montreal, Canada.

Brown, N. (2004). *Evaluation of Multidisciplinary Optimization (MDO) Techniques Applied to a Reusable Launch Vehicle* (Tech. Rep.). Atlanta, GA: Georgia Institute of Technology.

Celaya, J., Saxena, A., & Saha, S. (2011). Prognostics of Power MOSFETs under Thermal Stress Accelerated Aging using Data-Driven and Model-Based Methodologies. In *Annual conference of the prognostics and health management society* (pp. 1–10). Montreal, Canada.

Clarich, A., & Pediroda, V. (2004). A Competitive Game Approach for Multi-Objective Robust Design Optimization. In *1st intelligent systems technical conference.* Chicago, IL.

Cramer, E. J., Dennis, J. E. J., Frank, P. D., Shubin, G. R., & Lewis, R. M. (1993). Problem Formulation for Multidisciplinary Optimization. In *Aiaa symposium on multidisciplinary design optimization* (Vol. 4). SIAM. doi: 10.1137/0804044

Daigle, M., & Roychoudhury, I. (2010). Qualitative Event-Based Diagnosis: Case Study on the Second International Diagnostic Competition. In (pp. 1–8).

Das, A., & Chakrabarti, B. (2005). *Quantum Annealing and Related Optimization Methods (Lecture Notes in Physics).* Springer.

Denney, E., & Pai, G. (2012). Perspectives on Software Safety Case Development for Unmanned Aircraft. In *Ieee/ifip international conference on dependable systems and networks (dsn).* Boston, MA.

Driessen, B., & Kwok, K. (1998). A multiobjective dynamic programming approach to constrained discrete-time optimal control. In *Proceedings of the 1998 american control conference. acc (ieee cat. no.98ch36207)* (Vol. 5, pp. 3077–3083). American Autom. Control Council. doi: 10.1109/ACC.1998.688424

Edwards, D., Orchard, M. E., Tang, L., Goebel, K., & Vachtsevanos, G. (2010). Impact of Input Uncertainty on Failure Prognostic Algorithms : Extending the Remaining Useful Life of Nonlinear Systems. In *Annual conference of the prognostics and health management society* (pp. 1–7). Portland, OR.

Haddad, G., Sandborn, P., & Pecht, M. (2011a). A Real Options Optimization Model To Meet Availability Requirements For Offshore Wind Turbines. In *Mfpt* (pp. 1–12). Virginia Beach, VA.

Haddad, G., Sandborn, P., & Pecht, M. (2011b, June). Using real options to manage condition-based maintenance enabled by PHM. In *2011 ieee conference on prognostics and health management* (pp. 1–7). Denver, CO: Ieee. doi: 10.1109/ICPHM.2011.6024318

Hestenes, M. R. (1969, November). Multiplier and gradient methods. *Journal of Optimization Theory and Applications*, *4*(5), 303–320. doi: 10.1007/BF00927673

Hogge, E., Quach, C., Vazquez, S., & Hill, B. (2011). *A Data System for a Rapid Evaluation Class of Subscale Aerial*

Vehicle, NASA/TM2011-217145 (Tech. Rep. No. May). Hampton, VA: NASA Langley Research Center.

Hussein, M. L., & Abo-Sinna, M. A. (1993, November). Decomposition of multiobjective programming problems by hybrid fuzzy-dynamic programming. *Fuzzy Sets and Systems*, *60*(1), 25–32. doi: 10.1016/0165-0114(93)90286-Q

Iyer, N., Goebel, K. F., & Bonissone, P. (2006). Framework for Post-Prognostic Decision Support. *2006 IEEE Aerospace Conference*, 1–10. doi: 10.1109/AERO.2006.1656108

Kawaguchi, J., Uesugi, K., & Fujiwara, A. (2003, January). The MUSES-C mission for the sample and returnits technology development status and readiness. *Acta Astronautica*, *52*(2-6), 117–123. doi: 10.1016/S0094-5765(02)00146-7

Kim, H. M. (2001). *Target Cascading in Optimal System Design*. Unpublished doctoral dissertation, The University of Michigan.

Kim, H. M., Michelena, N. F., Papalambros, P. Y., & Jiang, T. (2003). Target Cascading in Optimal System Design. *Journal of Mechanical Design*, *125*(3), 474. doi: 10.1115/1.1582501

Lewis, K., & Mistree, F. (1997). Modeling interactions in multidisciplinary design: A game theoretic approach. *AIAA Journal*, *35*(8), 1387–1392.

Liao, L. (2002). Adaptive differential dynamic programming for multiobjective optimal control. *Automatica*, 1–27.

Marston, M. C. (2000). *Game based design: a game theory based approach to engineering design* (Vol. 0). Unpublished doctoral dissertation, Georgia Institute of Technology.

Mosterman, P. J., & Biswas, G. (1999). Diagnosis of continuous valued systems in transient operating regions. *IEEE Transactions on Systems Man and Cybernetics Part A Systems and Humans*, *29*(6), 554–565. doi: 10.1109/3468.798059

Narasimhan, S. (2007). HyDE A General Framework for Stochastic and Hybrid Model-based Diagnosis. In *Proceedings of the international workshop on the principles of diagnosis.* Nashville, TN.

NTSB. (1989). *Aircraft Accident Report PBSO-910406 NTSB/AAR-SO/06* (Tech. Rep.). Washington, D.C..

Pereira, E. B., Galvao, R. K. H., & Yoneyama, T. (2010, July). Model Predictive Control using Prognosis and Health Monitoring of actuators. In *2010 ieee international symposium on industrial electronics* (pp. 237–243). IEEE. doi: 10.1109/ISIE.2010.5637571

Rao, S. S. (2009). *Engineering Optimization*. Hoboken, NJ, USA: John Wiley and Sons, Inc. doi: 10.1002/9780470549124

Rao, S. S., & Freiheit, T. I. (1991). A Modified Game Theory Approach to Multiobjective Optimization. *Journal of Mechanical Design*, *113*(September), 286–291.

Renaud, G., & Shi, G. (2002). Evaluation and implementation of multidisciplinary design optimization strategies. In *Congress of the international council of the aeronautical sciences (icas)* (pp. 1–10).

Renaud, J. E., & Gabriele, G. A. (1993). Improved Coordination in Nonhierarchic System Optimization. *AIAA Journal*, *31*(12), 2367–2373. doi: 10.2514/3.11938

Roth, B. D. (2008). Enhanced collaborative optimization: application to an analytic test problem and aircraft design. In *Analysis and optimization conference* (pp. 1–14).

Roth, B. D., & Kroo, I. M. (2008). Enhanced Collaborative Optimization: A Decomposition-Based Method for Multidisciplinary Design. In *34th design automation conference,* (Vol. 2008, pp. 927–936). Brooklyn, NY: ASME. doi: 10.1115/DETC2008-50038

Saha, B., & Goebel, K. F. (2009). Modeling Li-ion Battery Capacity Depletion in a Particle Filtering Framework. In *Annual conference of the prognostics and health management society* (pp. 1–10). San Diego, CA: PHM Society.

Saha, B., Koshimoto, E., Quach, C. C., Hogge, E., Strom, T. H., Hill, B. L., … Goebel, K. F. (2011). Battery health management system for electric UAVs. *2011 IEEE Aerospace Conference*, 1–9. doi: 10.1109/AERO.2011.5747587

Sellar, R., Batill, S. M., & Renaud, J. E. (1996). Response Surface Based, Concurrent Subspace Optimization For Multidisciplinary System Design. In *34th aiaa aerospace sciences meeting and exhibit.* Reno, NV.

Shan, S., & Wang, G. G. (2009, August). Survey of modeling and optimization strategies to solve high-dimensional design problems with computationally-expensive black-box functions. *Structural and Multidisciplinary Optimization*, *41*(2), 219–241. doi: 10.1007/s00158-009-0420-2

Sobieszczanski-Sobieski, J., Agte, J. S., & Sandusky, R.

(1998). Bi-Level Integrated System Synthesis. *AIAA Journal*, *38*(August), 164–172.

Sobieszczanski-Sobieski, J., Emiley, M. S., Agte, J. S., & Sandusky, R. (2000). Advancement of Bi-Level System Synthesis (BLISS). In *Aerospace sciences meeting and exhibit.* Reno, NV.

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* (pp. 1–11).

Tang, L., Kacprzynski, G. J., Goebel, K., Reimann, J., Orchard, M. E., Saxena, A., & Saha, B. (2007). Prognostics in the Control Loop. In *Working notes of 2007 fall aaai symposium* (pp. 128–135).

Tosserams, S., Etman, L. F. P., Papalambros, P. Y., & Rooda, J. E. (2005). Augmented Lagrangian Relaxation for Analytical Target Cascading. In *6th world congress on structural and multidisciplinary optimization* (pp. 1–11).

Vincent, T. L. (1983). Game Theory as a Design Tool. *ASME Journal Of Mechanisms, Transmissions, and Automation in Design*, *105pp*, 165–170.

Wolpert, D. H., Strauss, C. E. M., & Rajnarayan, D. (2006). Advances in Distributed Optimization Using Probability Collectives. *Advances in Complex Systems*, *9*(4), 383–436.