Infotech@Aerospace 2012
19 - 21 June 2012, Garden Grove, California

AIAA 2012-2403

A Mission Based Fault Reconfiguration Framework for Spacecraft Applications

Ali Nasir[*] and Ella M. Atkins.[†], Ilya V. Kolmanovsky[‡]
University of Michigan, Ann Arbor, Michigan, 48105

Abstract

We present a Markov Decision Process (MDP) framework for computing post-fault reconfiguration policies that are optimal with respect to a discounted cost. Our cost function penalizes states that are unsuitable to achieve the remaining objectives of the given mission. The cost function also penalizes states where the necessary goal achievement actions cannot be executed. We incorporate probabilities of missed detections and false alarms for a given fault condition into our cost to encourage the selection of policies that minimize the likelihood of incorrect reconfiguration. To illustrate the implementation of our proposed framework, we present an example inspired by the Far Ultraviolet Spectroscopic Explorer (FUSE) spacecraft with a mission to collect scientific data from 5 targets. Using this example, we also demonstrate that there is a design tradeoff between safe operation and mission completion. Simulation results are presented to illustrate and manage this tradeoff through the selection of optimization parameters.

Nomenclature

F = Vector of fault flags ($F = \{f^1, f^2, \ldots, f^{m1}\}$).
P = Vector of probabilities of correctness of fault flags ($P = \{p^1, p^2, \ldots, p^{m1}\}$).
$P(MD^i)$ = Probability of missed detection for fault flag i. $P(MD^i) = (1 - p^i)(1 - f^i)$
$P(FA^i)$ = Probability of false alarm for fault flag i. $P(FA^i) = (1 - p^i)f^i$.
O = Vector of abstracted sensor observations ($O = \{o^1, o^2, \ldots, o^{m2}\}$).
sw = Scalar index of system hardware configuration of the spacecraft.
c = Scalar index of active control law for the spacecraft.
A = Vector of binary flags indicating active/inactive mission related actions ($A = \{a^1, a^2, \ldots, a^{n1}\}$).
B = Vector of binary flags indicating complete/incomplete mission objectives ($B = \{b^1, b^2, \ldots, b^{n2}\}$).
S = Set of MDP states ($S = \{s_1, s_2, \ldots, s_N\}$). Where $s_i = \{A_i, B_i, F_i, P_i, sw_i, c_i\}$ or $s_i = \{A_i, B_i, F_i, O_i, sw_i, c_i\}$.
M = Set of actions for MDP ($M = \{\mu_1, \mu_2, \ldots, \mu_k\}$).
$R(s_i)$ = Reward function for state s_i.
$V(s_i)$ = Value function for state s_i.
$T(s_j \mid \mu_k, s_i)$ = Probability of transitioning from state s_i to s_j by executing action μ_k.
γ = discount factor for reward computation.
$G(s_i)$ = Penalty function for reconfiguration under uncertain detection flags.
α, β, λ = Positive constant weighting factors.
θ_1 = Probability of success of the switching reconfiguration action.
θ_2 = Probability of success of the control law reconfiguration action.

I Introduction

Today's space missions are increasingly sophisticated which is in part due to improvements in onboard sensing and computing capabilities. One of the critical challenges for autonomous or semi-autonomous space missions is

[*] Graduate Student, Aerospace Engineering, Ann Arbor, MI 48109, email: techibro@umich.edu, AIAA student member.
[†] Associate Professor, Aerospace Engineering, Ann Arbor, MI 48109, email: ematkins@umich.edu, Associate Fellow.
[‡] Professor, Aerospace Engineering, Ann Arbor, MI 48109, email: ilya@umich.edu, AIAA member.

reliable, fault tolerant mission execution. This requires a combination of fault detection and subsequent reconfiguration. Because fault detection and reconfiguration are inherently coupled, the system must additionally decide when it is better to reconfigure versus maintain the current configuration in a possibly degraded capacity. It is also important to determine what type of reconfiguration is optimal given the estimated likelihood that the fault detection report is accurate.

Previously in fault reconfiguration, researchers have taken into account the uncertainty of fault detection. For example Rago et. al.[1] have proposed a fault tolerant control scheme where the post-detection control law is a weighted sum of the stabilizing controllers for different failure modes. In this formulation, the weight on each control law depends upon the probability of the corresponding failure as predicted by the detection scheme. Also, Abu Bakar and Veres[2] have proposed a multi-agent fault tolerant planning architecture where the reconfiguration agent iterates on various reconfiguration actions while learning from iterations until the response of the system is satisfactory.

Our goal in this paper is to devise a reconfiguration scheme that not only takes into account the uncertainty in fault detection and the possibility of failure of a reconfiguration action, but also incorporates the mission objectives and current policy execution state into the decision-making process. The unique features of our approach, as compared to the existing approaches are formulation of the reconfiguration problem as a Markov Decision Process (MDP)[3] and the introduction of an explicit tradeoff between spacecraft safety and importance of mission completion. Each state of the MDP contains information about detected faults, current status of the mission in terms of its remaining objectives, probabilities of correctness of fault detection, status of the mission related actions in terms of in-progress versus not-in-progress, and the current status of all reconfiguration options. We use the value iteration approach[3] to compute an optimal reconfiguration policy that maximizes the expected discounted performance reward which is a function of our MDP state. The resulting policy provides an optimal reconfiguration action for each state of the MDP. Specifically, the policy prescribes the reconfiguration action as a function of a state that contains information about fault decision, probability of correctness of the fault decision, status of the current mission, each ongoing mission related action, and status of reconfiguration options. Given that MDP-based deliberation is computationally-intensive and the complexity grows exponentially with the size of the state space, we must assume that the probability distribution of fault decisions can be represented with a tractable set of reachable states for time horizons under consideration.

For example, consider a 1DOF spacecraft with a reaction wheel and associated electronics. For this system, we can construct a Bayes Network[4] for the fault probabilities of the electronics and the wheel based on abstracted sensor readings that contain information about the possible failure modes for the wheel and electronics. For every possible value of the evidence, corresponding failure probabilities are computed based on the Bayes Net. Since the evidence contains information about the failure modes, there is a finite set of possible values for the evidences and hence there are finitely many values for the probabilities of failures computed from this evidence. Since probabilities of correctness of fault decisions are computed from probabilities of failures and the value of the fault decision flags, this implies that the probability distribution of fault decisions can be represented by a finite set of discrete or symbolic values. This guarantees finite state space for the MDP in which state contains information about probabilities of correctness of fault decisions.

Below, we describe the problem, provide some background on MDPs and value iteration, and then present our MDP-based solution framework. In Section III we present an implementation example inspired by FUSE mission[5]. In Section IV we present simulation results and discuss the tradeoff between safe operations versus emphasis upon mission completion. Finally, we present conclusions and future work in section V.

II Problem Formulation and Solution Approach

A. Problem Statement

We develop a framework for constructing a reconfiguration strategy that is optimal with respect to minimizing the possibility of incorrect reconfiguration, maximizing the possibility of achieving the remaining mission objectives, and maximizing the possibility of completion of in-progress mission related actions, while accounting for the possibility of failure of the reconfiguration action. Our assumptions are stated below:

A1. The spacecraft is on a mission that can be decomposed into a set of mission objectives.

A2. A status vector for mission objectives, assigned achieved and not-achieved values, is available as an input to the reconfiguration algorithm.

A3. A status vector for mission-related actions, assigned in-progress and not-in-progress values, is available as an input to the reconfiguration algorithm.

A4. The system has an on-board fault detector that provides fault information.

A5. Status of the fault detection decisions from the fault detector is available as an input to the reconfiguration algorithm.

A6. Either the probabilities of correctness of fault decisions are available to the reconfiguration algorithm, or the abstracted sensor readings are available from which the probabilities of correctness can be calculated using joint failure probability distribution model, i.e. Bayes net.

A7. Reconfiguration actions are executed instantaneously i.e. fault decisions, status of the mission objectives, status of mission related actions, and probabilities of correctness of fault decisions do not change during the execution of a reconfiguration action.

B. MDP Background

An MDP policy is constructed using a discrete stochastic dynamic programming algorithm such as value iteration or policy iteration[3]. In this section, we present background on value iteration. Value iterations are applied to the optimal control problem to maximize expected discounted reward function of the form

$$V^{Pol}(s) = E\left[\sum_{t=0}^{\infty} \gamma^t R(s^t) \mid Pol, s^0 = s\right] \tag{1}$$

Here, s^t represents state after t actions (s^t is a random variable). *Pol* represents the policy according to which the actions have been taken. V is the expected discounted reward function of states of the Markov chain (also called the value function of the state). The discount factor γ ($\gamma \in [0,1)$), indicates that future rewards have lower value. Note that we represent the case where reward R is only the function of state and not action. We also assume that R is bounded from above and below. The policy that selects the optimal action may be found as

$$Pol^*(s_i) \in \arg\max_k \left(\sum_{j \in S} T(s_j \mid \mu_k, s_i) V(s_j) \right) \tag{2}$$

There is a direct relationship between the value of a state and the values of all the states that can be reached from that state in a single optimal action. This relationship can be expressed using the Bellman equation:

$$V_{t+1}(s_i) = R(s_i) + \max_k \left(\sum_{j \in S} \gamma T(s_j \mid \mu_k, s_i) V_t(s_j) \right) \tag{3}$$

where, $V_{t+1}(s_i)$ is the value of state s_i at iteration $t+1$, $R(s_i)$ is the immediate reward of state s_i, *and* $T(s_j \mid \mu_k, s_i)$ is the probability of transitioning from state s_i to s_j by executing action μ_k.

Value iterations converge exponentially fast and one can bound the number of iterations (*Itr*) to reach an error bound of ε as:

$$Itr = \left[\log\left(\frac{2R_{\max}}{(1-\gamma)\varepsilon} \right) / \log\left(\frac{1}{\gamma} \right) \right] \tag{4}$$

Here R_{max} is upper bound on reward function and ε is the required tolerance of the solution satisfying

$$\left\| V_{t+1}(\varsigma_i) - V(\varsigma_i) \right\| < \varepsilon, \forall i \tag{5}$$

Inequality (5) is ensured by

$$\left\| V_{t+1}(\varsigma_i) - V_t(\varsigma_i) \right\| < \varepsilon \left(\frac{1-\gamma}{\gamma} \right) \tag{6}$$

Note that the computational complexity of value iteration is of the order O(N^2k), where N is the number of states and k is the number of actions in the MDP.

Result[3]: Under the assumption of bounded reward function and $\gamma \in [0,1)$, the value functions iterations in equation (3) converge to a unique solution. The solution of equation (3) achieves the maximum value of right hand side in equation (1). If the policy is calculated using (2) with the solution of (3), it will be optimal with respect to (1).

C. MDP Formulation

We present two possible formulations for the states of the MDP. One of the formulations include probabilities of correctness of fault decision flags as part of MDP states whereas the other formulation includes abstracted sensor readings from which the probabilities of correctness of fault flags are calculated using Bayes net as described in the introduction section. Both formulations are equivalent to each other although depending upon specifics of the application; one may yield lesser number of states than the other.

The formulation with probabilities of correctness of fault flags included as part of state can be represented as

$$\begin{aligned}
&S = \{s_1, s_2, s_3, ..., s_N\} \\
&s_i = \{a_i^1, a_i^2, ..., a_i^{n1}, b_i^1, b_i^2, ..., b_i^{n2}, f_i^1, f_i^2, ..., f_i^{m1}, p_i^1, p_i^2, ..., p_i^{m1}, sw_i, c_i\} \\
&OR \\
&s_i = \{A_i, B_i, F_i, P_i, sw_i, c_i\} \\
&i \in \{1, 2, ..., N\}
\end{aligned} \tag{7}$$

where A is vector of mission objective flags representing whether or not a certain objective has been achieved, B is a vector of mission related action flags indicating which of the actions are in progress or not-in-progress at the moment, F is a vector of fault flags generated by fault detection mechanism, and P is a vector of probabilities of correctness of the fault flags in F. Note that there is a one-to-one relation between the probabilities of false alarms/missed detections and the probabilities of correctness of fault decisions e.g. for a fault j in state i, $P(FA^j{}_i) = (1 - p^j{}_i)f^{\,j}{}_i$ and $P(MD^j{}_i) = (1 - p^j{}_i)(1 - f^{\,j}{}_i)$. The discrete variable sw represents the current system hardware configuration and c is a discrete variable representing the currently active control law for the system. Note that from assumption A.6 we can construct a Bayes net for the spacecraft faults which can be used to calculate probabilities of failure modes based on which we can calculate the probabilities of missed detections and false alarms. A further implication of A.6 is that there are finitely many values for the elements of P as explained in the introduction section.

The second state formulation can be represented as

$$\begin{aligned}
&S = \{s_1, s_2, s_3, ..., s_N\} \\
&s_i = \{a_i^1, a_i^2, ..., a_i^{n1}, b_i^1, b_i^2, ..., b_i^{n2}, f_i^1, f_i^2, ..., f_i^{m1}, o_i^1, o_i^2, ..., o_i^{m2}, sw_i, c_i\} \\
&OR \\
&s_i = \{A_i, B_i, F_i, O_i, sw_i, c_i\} \\
&i \in \{1, 2, ..., N\}
\end{aligned} \tag{8}$$

Note that this formulation is exactly the same except for the replacement of detection correctness probability vector P with abstracted sensor-based observation vector O. This representation is equivalent to the earlier one because O contains information that is combined with information in vector F (along with sw, c, and A) to compute the correctness probabilities of the detection from the Bayes Net (Assumption A.6).

The actions for both formulations are represented as

$$M = \{sw_1, sw_2, ..., sw_{m3}, c_1, c_2, ..., c_{m4}, NOOP\} \tag{9}$$

where sw_i represents the selection of a particular system hardware configuration (i) among $m3$ possible system hardware configurations and c_j represents selection of a particular control law (j) among $m4$ possible control laws. We also have an option of doing no reconfiguration in the form of a "no operation" ($NOOP$) action.

The reward function is the most important part of this formulation because all optimization is with respect to this function. We define our reward function for the state formulation in (7) as

$$R(s_i) = \lambda \exp\left\{ \begin{array}{l} -\sum_{k=1}^{n1} \alpha_k I\left(s_i \in A_k^{critical}\right) a_i^k - \sum_{k=1}^{n2} \beta_k I\left(s_i \in B_k^{critical}\right)\left(1 - b_i^k\right) \\ -\sum_{k=1}^{m1} G\left(f_i^k, p_i^k, sw_i, c_i\right) \end{array} \right\} \tag{10}$$

where α, β, and λ are positive constants. I is an indicator function ($I(x) = 1$ when x is true and 0 otherwise). $A_k{}^{critical}$ is a subset of state space where activation of a certain mission related action (a^k) is critical or undesired. $B_k{}^{critical}$ is a subset of the state space containing states that can lead to inability to achieve the mission objective k or the states that can make it difficult to achieve the mission objective k. Also, G is a semi-positive-definite function of fault flags, their corresponding correctness probabilities, and current switching and control configuration. G should be selected such that it is zero if the switching and control configuration is optimal (in some sense e.g. safe operation of the spacecraft) for the current fault flags and probabilities of correct detection. G should be positive otherwise, with highest value for the worst possible configuration. Note that the reward function has three main terms inside exponential. The first term penalizes all the states where the active mission-related action is undesirable, i.e., the states that we would like to avoid when we know a particular mission-related action is active. The second term penalizes being in states which are undesirable given the unachieved objectives. The third and final term penalizes being in states where the current configuration is suboptimal given the current fault flags and their probabilities of correctness. The reward function is chosen to be an exponential to make it non-negative.

The reward function for states for the second formulation with observation vector *O* can be represented as

$$R(s_i) = \lambda \exp\left\{ \begin{aligned} & -\sum_{k=1}^{n1} \alpha_k I\left(s_i \in A_k^{critical}\right) a_i^k - \sum_{k=1}^{n2} \beta_k I\left(s_i \in B_k^{critical}\right)\left(1 - b_i^k\right) \\ & -\sum_{k=1}^{m1} \overline{G}\left(f_i^{\,k}, O_i^k, sw_i, c_i\right) \end{aligned} \right\} \tag{11}$$

where $O_i^{\,k}$ is the subset of O_i that is used to calculate $p_i^{\,k}$. Terms of (11) map to analogous terms in (10).

Two cases of transition probability specifications are provided for situations involving instantaneous and non-instantaneous reconfiguration actions. Both of the presented cases are applicable to both the MDP state formulations of (7) and (8). In the first case, we assume that the actions are instantaneous with respect to the changing fault flags and other conditions (Assumption A.7). Therefore, each action could simply result either in the desired post-execution state or the same state from which it is executed depending upon the probability of failure of the reconfiguration action, indicating a situation where the action did not change the state in the desired manner but also did not have side effects on other state elements. The transition probability function can therefore be represented as

$$T(s_j \mid s_i, \mu_k) = \begin{cases} \theta_1 & if : \left(\left\{s_j \setminus sw_j\right\} = \left\{s_i \setminus sw_i\right\}\right) \wedge sw_j = \mu_k \\ \theta_2 & if : \left(\left\{s_j \setminus c_j\right\} = \left\{s_i \setminus c_i\right\}\right) \wedge c_j = \mu_k \\ 1-\theta_1 & if : s_j = s_i \wedge sw_j = \mu_k \\ 1-\theta_2 & if : s_j = s_i \wedge c_j = \mu_k \end{cases} \tag{12}$$

$$\mu_k \in M$$

where θ_1 and θ_2 are probabilities of the success of switching and control law reconfiguration actions respectively. Also, $x \setminus y$ is the set difference operation, returning elements of set *x* that are not in *y*.

In the second case, we remove restrictions on transition changes to state features during execution of reconfiguration actions, i.e. we discard assumption A.7. The resulting transition probability function is presented in Equation (13). Note that although the probabilities of each transition type are shown separately, any combination of the transitions can happen in any state. For example, given a reconfiguration action applied, the reconfiguration may fail and on the other hand, one of the unachieved mission objectives may become achieved. Also, the sensor output could change hence changing the fault flag correctness probabilities, or the fault detection scheme may decide to change fault flags based on new sensor information etc. Note that the value of fault flags and probability of failure depend upon sensor readings, but sensor readings are independent of fault flag values. Also, the probabilities of failures are conditionally independent of the values of fault flags given the sensor readings.

$$T(s_j \mid s_i, \mu_k) = \begin{cases} \theta_1 & Transition(sw_i) \\ \theta_2 & Transition(c_i) \\ \sum_{l=1}^{n1} \varphi 1_l & Transition(A_i) \\ \sum_{l=1}^{n2} \varphi 2_l & Transition(B_i) \\ \sum_{l=(1,1)}^{(m1,m1)} \varphi 3_l & Transition(P_i, F_i), OR, Transition(O_i, F_i) \\ 1-\theta_1 - \sum_{l=1}^{n1} \varphi 1_l - \sum_{l=1}^{n2} \varphi 2_l - \sum_{l=1}^{m1} \varphi 3_l & NoTransition(\mu_k = sw_j) \\ 1-\theta_2 - \sum_{l=1}^{n1} \varphi 1_l - \sum_{l=1}^{n2} \varphi 2_l - \sum_{l=1}^{m1} \varphi 3_l & NoTransition(\mu_k = c_j) \end{cases} \tag{13}$$

For simplicity in the following case study, we assume the case of instantaneous reconfiguration actions is true and also we will assume that the states are represented with vector *O* instead of *P*. Representation with *P* is similar.

III Baseline Spacecraft Case Study

Our case study is inspired by Far Ultraviolet Spectroscopic Explorer (FUSE) mission launched in June 1999[5]. FUSE had four reaction wheels, two sets of three degree of freedom gyroscopes, three magnetic torqueing bars, three-axis magnetometers, course sun sensors, a fine error sensor, and a payload of 4 co-aligned prime focus telescopes and Rowland spectrographs with micro-channel plate detectors to observe light rays in the ultraviolet spectrum ranging 905 to 1187 Angstroms. FUSE suffered failures in three out of four reaction wheels and both of its sets of gyroscopes also failed[6, 7], offering a compelling real-world example to study with our fault reconfiguration framework. In response to the failures, FUSE's control laws and gyroscopes were reconfigured to re-establish acceptable operational performance, but since these reconfigurations were performed manually from the ground station, substantial mission time was sacrificed with the spacecraft in safe mode while awaiting reconfiguration instructions. . Here we show how our proposed framework could have enabled the FUSE spacecraft to automatically reconfigure itself to use alternate control laws in response to the encountered failures. Below, we first describe a model for FUSE in our MDP formulation followed by discussion of results from the application of MDP policies generated from this model in cases where failures are experienced.

The states of the MDP for FUSE spacecraft reconfiguration can be defined as

$$\begin{aligned}
&s_i = \left\{A_i, B_i, F_i, O_i, sw_i, c_i\right\} \\
&where, \\
&A_i \in \left\{0,1,2\right\} \\
&B_i = \left[b_i^1, b_i^2, b_i^3, b_i^4, b_i^5\right] \\
&F_i = \left[f_i^1, f_i^2, ..., f_i^{10}\right] \\
&O_i = \left[o_i^1, o_i^2, ..., o_i^{10}\right] \\
&sw_i = \left[sw_i^{roll}, sw_i^{pitch}, sw_i^{yaw}\right] \\
&c \in \left\{1, 2, ..., 14\right\}
\end{aligned} \tag{14}$$

In above equation, we have assumed 5 data collection targets. Also, *A* represents whether the spacecraft is executing an attitude maneuver (*A* = 1), collecting scientific data (*A*=2), or doing nothing (*A*=0). *B* includes binary flags indicating status of the data collection objectives. *F* represents fault flags generated by the detection scheme for six gyroscopes and four reaction wheels. *O* represents processed sensor data that represents fail/not fail status of the gyroscopes and the reaction wheels. This data is used to calculate the probabilities of false alarms and missed detections for the flags in *F*. Vector *sw* represents which of the gyroscopes are in use along each of the axis of rotation. The active control law is represented by *c*. We assume four three-wheel control laws, six two-wheel control laws, and four one-wheel control laws. The one-wheel control laws are assumed to be assisted by magnetic torqueing bars.

With the above definition of state, the total number of states for the MDP is 11,274,289,152 and is too large to handle computationally. Consequently, we simplify our problem by observing that the selection of gyroscopes can be separated from mission objectives and mission related actions. This is rationalized on the basis that the angular velocities can be estimated using the dynamic model of the spacecraft and information about its attitude although the pointing accuracy may be reduced. Thus the reconfiguration of gyroscopes can be separated from the reconfiguration of the control laws. The subsystem for control law reconfiguration then becomes

$$\begin{aligned}
&s_i = \left\{A_i, B_i, F_i, O_i, c_i\right\} \\
&where, \\
&A_i \in \left\{0,1,2\right\} \\
&B_i = \left[b_i^1, b_i^2, b_i^3, b_i^4, b_i^5\right] \\
&F_i = \left[f_i^1, f_i^2, ..., f_i^4\right] \\
&O_i = \left[o_i^1, o_i^2, ..., o_i^4\right] \\
&c \in \left\{1,2,...,14\right\}
\end{aligned} \tag{15}$$

The size of this state space is now 344,064 which is manageable with currently-available desktop or laptop computers. The states for gyroscope reconfiguration are then defined as

$$\begin{aligned}
&s_i = \left\{F_i, O_i, sw_i\right\} \\
&where, \\
&F_i = \left[f_i^1, f_i^2, ..., f_i^6\right] \\
&O_i = \left[o_i^1, o_i^2, ..., o_i^6\right] \\
&sw_i = \left[sw_i^{roll}, sw_i^{pitch}, sw_i^{yaw}\right]
\end{aligned} \tag{16}$$

The size of the state space in (16) is 32,768 which is easily manageable. The actions of each system are related to selection of a control law for the control law reconfiguration MDP, and selection of a gyroscope for the gyroscope reconfiguration MDP. Initially, we assume that all these selections are deterministic; therefore, there are no transition probabilities involved. The reward function for the control law reconfiguration MDP is defined as,

$$\begin{aligned}
&R\left(s_i\right) = \lambda \exp\left\{\begin{matrix} -\sum_{k=1}^{5} \beta_k IsMissionCritical(c_i, b_i^k) - \alpha IsActionCritical(c_i, A_i) \\ -G\left(F_i, O_i, c_i\right) \end{matrix}\right\} \\
&where, \\
&IsMissionCritical(c_i, b_i^k) = \begin{cases} 0 & if: b_i^k = 1, or, c_i \in \left\{1,2,3,4\right\} \\ 1 & if: b_i^k = 0, and, c_i \in \left\{5,6,..10\right\} \\ 2 & if: b_i^k = 0, and, c_i \in \left\{11,12,13,14\right\} \end{cases} \\
&IsActionCritical(c_i, A_i) = \begin{cases} 0 & if: A_i = 0, or, c_i \in \left\{1,2,3,4\right\} \\ 1 & if: A_i > 0, and, c_i \in \left\{5,6,..10\right\} \\ 2 & if: A_i > 0, and, c_i \in \left\{11,12,13,14\right\} \end{cases}
\end{aligned} \tag{17}$$

In above equation, the *IsMissionCritical* function determines the penalty of using less capable control law for each of the mission objectives. In our case study, we base this function on the number of reaction wheels in-use for the active control law. Precisely, we set β_k = 1 for all k except for β_3 = 5 to indicate that the objective 3 has more importance than the other objectives. We set the *IsMissionCritical* function to return 0 if c uses 3 wheels, 1 if c uses two wheels, and 2 if c uses 1 wheel. Similarly, we set α = 2 and define *IsActionCritical* to return 0 if either A = 0 or c uses 3 wheels, 1 if A > 0 and c uses two wheel, and 2 if A > 0 and c uses 1 wheel. As for the function G, we set up the penalties for each control law such that there is a penalty of 2.5 for each in-use wheel with the fault flag in F activated. There also is a penalty of 2.5 for each not-in-use wheel with the fault flag in F turned off. These penalties are added along with the additional penalties of value 1 corresponding to flags in O. For example, if the flags in F indicate failure of reaction wheel 2 (i.e. F = {0 1 0 0}) and the flags in O indicate failure of reaction wheels 1 and 3 (i.e. O = {0 1 0 1}), the value returned by G for control law using the wheels 1, 2, and 3 would be 8. However, if c uses three wheels and all of them are deemed healthy by both F an O, then there is no penalty for not using the fourth wheel. Details of calculation of G function for each of the control laws is presented in Table 1.

Table 1: Calculation of *G*(*F, O, c*) function

In use Control law	Function for calculating value of G

<table>
<tr><td></td><td><i>F</i>(i) = 1 if i^{th} wheel is faulty
<i>O</i>(i) = 0 if i^{th} wheel is faulty</td></tr>
<tr><td><i>c</i> = 1, wheel(s) in use {1, 2, 3}</td><td><code>(2.5*(F(1)+F(2)+F(3)) + (~O(1)+~O(2)+~O(3)) + (F(1)||F(2)||F(3)||~O(1)||~O(2)||~O(3))*(~F(4)*2.5 + O(4)))</code></td></tr>
<tr><td><i>c</i> = 2, wheel(s) in use {1, 2, 4}</td><td><code>(2.5*(F(1)+F(2)+F(4)) + (~O(1)+~O(2)+~O(4)) + (F(1)||F(2)||F(4)||~O(1)||~O(2)||~O(4))*(~F(3)*2.5 + O(3)))</code></td></tr>
<tr><td><i>c</i> = 3, wheel(s) in use {1, 3, 4}</td><td><code>(2.5*(F(1)+F(3)+F(4)) + (~O(1)+~O(3)+~O(4)) + (F(1)||F(4)||F(3)||~O(1)||~O(4)||~O(3))*(~F(2)*2.5 + O(2)))</code></td></tr>
<tr><td><i>c</i> = 4, wheel(s) in use {2, 3, 4}</td><td><code>(2.5*(F(2)+F(3)+F(4)) + (~O(2)+~O(3)+~O(4)) + (F(4)||F(2)||F(3)||~O(4)||~O(2)||~O(3))*(~F(1)*2.5 + O(1)))</code></td></tr>
<tr><td><i>c</i> = 5, wheel(s) in use {1, 2}</td><td><code>(2.5*(F(1)+F(2)) + (~O(1)+~O(2)) + (~F(3)*2.5+~F(4)*2.5) + (O(3)+O(4)))</code></td></tr>
<tr><td><i>c</i> = 6, wheel(s) in use {1, 3}</td><td><code>(2.5*(F(1)+F(3)) + (~O(1)+~O(3)) + (~F(2)*2.5+~F(4)*2.5) + (O(2)+O(4)))</code></td></tr>
<tr><td><i>c</i> = 7, wheel(s) in use {1, 4}</td><td><code>(2.5*(F(1)+F(4)) + (~O(1)+~O(4)) + (~F(3)*2.5+~F(2)*2.5) + (O(3)+O(2)))</code></td></tr>
<tr><td><i>c</i> = 8, wheel(s) in use {2, 3}</td><td><code>(2.5*(F(2)+F(3)) + (~O(2)+~O(3)) + (~F(1)*2.5+~F(4)*2.5) + (O(1)+O(4)))</code></td></tr>
<tr><td><i>c</i> = 9, wheel(s) in use {2, 4}</td><td><code>(2.5*(F(2)+F(4)) + (~O(2)+~O(4)) + (~F(3)*2.5+~F(1)*2.5) + (O(3)+O(1)))</code></td></tr>
<tr><td><i>c</i> = 10, wheel(s) in use {3, 4}</td><td><code>(2.5*(F(3)+F(4)) + (~O(3)+~O(4)) + (~F(1)*2.5+~F(2)*2.5) + (O(1)+O(2)))</code></td></tr>
<tr><td><i>c</i> = 11, wheel(s) in use {1}</td><td><code>(2.5*(F(1)) + (~O(1)) + (~F(3)*2.5+~F(4)*2.5+~F(2)*2.5) + (O(3)+O(4)+O(2)))</code></td></tr>
<tr><td><i>c</i> = 12, wheel(s) in use {2}</td><td><code>(2.5*(F(2)) + (~O(2)) + (~F(3)*2.5+~F(4)*2.5+~F(1)*2.5) + (O(3)+O(4)+O(1)))</code></td></tr>
<tr><td><i>c</i> = 13, wheel(s) in use {3}</td><td><code>(2.5*(F(3)) + (~O(3)) + (~F(1)*2.5+~F(4)*2.5+~F(2)*2.5) + (O(1)+O(4)+O(2)))</code></td></tr>
<tr><td><i>c</i> = 14, wheel(s) in use {4}</td><td><code>(2.5*(F(4)) + (~O(4)) + (~F(3)*2.5+~F(1)*2.5+~F(2)*2.5) + (O(3)+O(1)+O(2)))</code></td></tr>
</table>

The reward function for the reconfiguration of gyroscopes can be defined with only the *G* term as in the reward function for the control law reconfiguration. Note that there are no constraints in reconfiguration options and no uncertainties in state transitions. Therefore, there is no need to compute the solution for infinite horizon or even finite horizon of length greater than 2 for our case study. This would have not been possible in the presence of constraints on the reconfiguration such as one-time on-off switches, non-instantaneous reconfiguration actions, temporarily irreversible reconfiguration options, and uncertainties in state transitions such as those presented in equation (13) etc. Since we have simplified the problem, this allows us to avoid value iteration and calculate the optimal policy for both MDPs simply by computing the following for each of the states.

$$Pol\left(s_i\right) = \arg\max_{\mu} R\left(s_j(\mu)\right) \tag{18}$$

IV Simulation Results

Table 2 shows a sample trajectory resulting from the optimal control law reconfiguration policy calculated for the FUSE case study using (18). The first column of Table 2 indicates the trajectory step number. The second column provides the current state information. The third column indicates the optimal policy for the current state. The fourth column of Table 2 represents an exogenous event (change in MDP state that is not caused by execution of any of the actions within the MDP). In step 1, we start with a state where all reaction wheels are healthy. The control law in-use is $c = 1$ that uses reaction wheels 1, 2, and 3 for attitude maneuvering and stabilization. Also, no mission-related action is in progress. The response of the optimal policy to this state is *NOOP*. In steps 2, 3 and 4, we show that the response of the policy to the change in mission-related action status and mission objectives completion status is *NOOP*. The response to an exogenous event in step i is shown by the policy in step $i + 1$. In step 5, we show that the response to an observed failure that is not yet called by the fault detection scheme is to change the control law such that the suspected failed wheel (wheel 2) is not in use anymore. In step 6, we show that the response to another suspected failure is *NOOP* instead of changing the control law. This is because mission objective 3 is modeled as very important and it is undesirable to use less than 3 wheels for completion of this objective. We deliberately model our MDP to elaborate this tradeoff between safety and mission completion preference. This tradeoff allows the reconfiguration policy to risk the spacecraft's safety for the sake of some really important mission objective such as collecting data from an event that occurs once in a very long time or is of critical importance. Note that, in step 8, the control law is changed in response to the completion of objective 3 because after the completion of critical objective, there is no need to jeopardize the safety of the spacecraft. Steps 9 and 10 show that the optimal policy does not select an undesirable one-wheel control law unless the evidence for the failure of other three wheels is strong.

Table 2: Sample Trajectory with Optimal Control Law Reconfiguration Policy

Trajectory Step #	State Information F→ Fault flags (1 if faulty) O→ Observation flags (0 if faulty)	Policy for the State	Exogenous Event(s)
1	$A = 0$, $B = [0\ 0\ 0\ 0\ 0]$, $F = [0\ 0\ 0\ 0]$, $O = [1\ 1\ 1\ 1]$, $c = 1$ (wheel(s) {1, 2, 3})	NOOP	A = 1
2	$A = 1$, $B = [0\ 0\ 0\ 0\ 0]$, $F = [0\ 0\ 0\ 0]$, $O = [1\ 1\ 1\ 1]$, $c = 1$ (wheel(s) {1, 2, 3})	NOOP	A = 2
3	$A = 2$, $B = [0\ 0\ 0\ 0\ 0]$, $F = [0\ 0\ 0\ 0]$, $O = [1\ 1\ 1\ 1]$, $c = 1$ (wheel(s) {1, 2, 3})	NOOP	B = [1 0 0 0 0], A = 0
4	$A = 0$, $B = [1\ 0\ 0\ 0\ 0]$, $F = [0\ 0\ 0\ 0]$, $O = [1\ 1\ 1\ 1]$, $c = 1$ (wheel(s) {1, 2, 3})	NOOP	B = [1 1 0 0 0], O = [1 0 1 1]
5	$A = 0$, $B = [1\ 1\ 0\ 0\ 0]$, $F = [0\ 0\ 0\ 0]$, $O = [1\ 0\ 1\ 1]$, $c = 1$ (wheel(s) {1, 2, 3})	$c = 3$ (wheel(s) {1, 3, 4})	O = [1 0 0 1]
6	$A = 0$, $B = [1\ 1\ 0\ 0\ 0]$, $F = [0\ 0\ 0\ 0]$, $O = [1\ 0\ 0\ 1]$, $c = 3$ (wheel(s) {1, 3, 4})	NOOP	F = [0 1 1 0]
7	$A = 0$, $B = [1\ 1\ 0\ 0\ 0]$, $F = [0\ 1\ 1\ 0]$, $O = [1\ 0\ 0\ 1]$, $c = 3$ (wheel(s) {1, 3, 4})	NOOP	B = [1 1 1 0 0]
8	$A = 0$, $B = [1\ 1\ 1\ 0\ 0]$, $F = [0\ 1\ 1\ 0]$, $O = [1\ 0\ 0\ 1]$, $c = 3$ (wheel(s) {1, 3, 4})	c = 7 (wheel(s) {1, 4})	O = [0 0 0 1]
9	$A = 0$, $B = [1\ 1\ 1\ 0\ 0]$, $F = [0\ 1\ 1\ 0]$, $O = [0\ 0\ 0\ 1]$, $c = 7$ (wheel(s) {1, 4})	NOOP	F = [1 1 1 0]
10	$A = 0$, $B = [1\ 1\ 1\ 0\ 0]$, $F = [1\ 1\ 1\ 0]$, $O = [0\ 0\ 0\ 1]$,	c = 14 (wheel(s) {4})	…………………

	c = 7 (wheel(s) {1, 4})		

A sample trajectory for gyroscope reconfiguration MDP is shown in Table 3. Here, faults in the redundant gyroscopes along the roll axis are represented by the first two components in *F* and *O* vectors, the faults in redundant gyroscopes along the pitch axis are represented by the third and the fourth components in *F* and *O* vectors, and the faults in redundant gyroscopes along the yaw axis are represented by the fifth and the sixth components in the *F* and *O* vectors. Vector *sw* represents which of the redundant gyroscopes is currently in-use. Step 2 in Table 3 shows the response to a suspected failure in a gyroscope along the roll axis. The optimal policy selects the redundant gyroscope option. In step 3, we show that when stronger evidence is available for the failure of the redundant gyroscope along the roll axis, the optimal policy refers back to the gyroscope that is less likely to have failed. Step 4 shows the response to a gyroscope failure about the pitch axis.

Table 3: Sample Trajectory with Optimal Gyroscope Reconfiguration Policy

Trajectory Step #	State Information F→ Fault flags (1 if faulty) O→ Observation flags (0 if faulty)	Policy for the State	Exogenous Event(s)
1	*F* = [0 0 0 0 0 0], *O* = [1 1 1 1 1 1], *sw* = [1 1 1]	NOOP	O = [0 1 1 1 1 1 1]
2	*F* = [0 0 0 0 0 0], *O* = [0 1 1 1 1 1], *sw* = [1 1 1]	sw = [2 1 1]	F = [0 1 0 0 0 0]

	F = [0 1 0 0 0 0],
	O = [0 1 1 1 1 1],
	sw = [2 1 1]
		sw = [1 1 1]	F = [1 1 1 0 0 0],
			O = [0 0 0 1 1 1]

4	*F* = [1 1 1 0 0 0], *O* = [0 0 0 1 1 1], *sw* = [1 1 1]	sw = [1 2 1]	……………

Note that, although Tables 2 and 3 do not show exactly the same failure pattern as in actual FUSE mission, it can be inferred from the presented results that if FUSE mission did have pre-defined post-failure control laws and the reconfiguration framework, there would have been no interruption in the mission execution due to failures. Here we do not claim that the reconfiguration framework presented in this paper would have been the best for FUSE mission as compared to other existing reconfiguration frameworks e.g. The Livingstone[23], but we do claim that our framework provides more detailed design by incorporating the tradeoff between mission completion and safety of the spacecraft. Furthermore, our framework incorporates most of the calculations off-line to determine the optimal policy before-hand whereas in the Livingstone optimal reconfiguration action is calculated on-line by using conflict directed best first search.

V Conclusions and Future Work

We have presented a framework for calculating optimal policies for mission-based fault reconfiguration. Our framework is robust in a sense that while calculating the optimal policy, we take into account not only the uncertainty in the detection of faults, but also the currently active mission related actions and remaining objectives of the mission. We have described a modeling strategy applicable for Markov Decision Process policy generation, and have identified some important mission completion versus safe operation tradeoffs through a case study inspired by the FUSE mission. The primary drawback of our framework is the computational complexity associated with value iteration over a large state-space. We have shown though our case study that it is possible to break larger problems into smaller tractable problems by using simple observations about the system. Another important aspect of our proposed framework is proper tuning of the parameters in the reward function. Although the tradeoff between mission completion preference and the safety of the spacecraft provides additional capability to the reconfiguration framework, it can also cause the reconfiguration policy to select dangerous options if the parameters in the reward function are not tuned carefully. In future work, we plan to pursue other methods to decompose spacecraft models into subsets of weakly coupled subsystem models and develop lower level reconfiguration strategies for each subsystem separately. Exploitation of hierarchical domain structure will enable reconfiguration policy development for distinct subsystems at lower levels pass key information from the lower level to the top-level reconfiguration system.

References

[1] Rago, C. et. al. "Failure Detection and Identification and Fault Tolerant Control using the IMM-KF with applications to the Eagle-Eye UAV." *Proceedings of the 37th IEEE Conference on Decision and Control,* Tampa, Florida, USA. December 1998.

[2] Abu Bakar, B. and Veres, S. "A multi-agent approach to integrated FDI and reconfiguration of autonomous systems." *In the proceedings of IASTED conference on Artificical intelligence and applications*, 2010. Innsbruck, Austria.

[3] Puterman, Martin L. *Markov Decision Processes: Discrete Stochastic Dynamic Programming* © John Wiley and Sons Inc. (1994).

[4] Russell, S. and Norvig, P., *Artificial Intelligence: A Modern Approach*, 2nd Edition, Prentice-Hall, Upper Saddle River, New Jersey 07458, 2005.

[5] Moos, H. W. et al., "Overview of the Far Ultraviolet Spectroscopic Explorer Mission," *ApJ 538*, pp. L1 – L6, 2000.

[6] Sahnow, David J., "Operations with the new FUSE observatory: three-axis control with one reaction wheel," *SPIE*, Vol. 6266, Paper 2, 2006.

[7] Kruk, J. W., et al, "FUSE In-Orbit Attitude Control with Two Reaction Wheels and No Gyroscopes," *Proc. SPIE 4854*, pp. 274-285, 2002.

[8] Halyo N. Broussard, J.R. Moerder, D.D. and Cahlayan, A.K. "Application of percomputed control laws in a reconfigurable aircraft flight control system." *Journal of Guidance, Control and Dynamics*, 1989.

[9] Wu, N. E. "Robust feedback design with optimized diagnostic performance." *IEEE transactions on automatic control*, Vol. 42(no.9), 1997.

[10] Van Hasselt, H. and Wiering, M.A. "Reinforcement Learning in Continuous Action Spaces." *IEEE International Symposium on approximate dynamic programming and reinforcement learning*, 2007, pp. 272-279.

[11] Thanapalan, K.K.T. Veres, S. M. et al. "Fault tolerant controller design to ensure operational safety in satellite formation flying." *In Decision and Control*, 2006 45th *IEEE Conference* on, pages 1562-1567, Dec. 2006.

[12] Bojinov, H. Casal, A. and Hogg, H. "Multiagent control of selfreconfigurable robots". *Artificial Intelligence*, Vol. 142:99, 2002.

[13] Britain, S. L. Gibb, A. J. and Roberts, C. "Automatic reconfiguration of a robotic arm using a multi-agent approach." *Proceedings of the Institution of Mechanical Engineers*, Part I: *Journal of Systems and Control Engineering*, 2008. 222(2): p. 127-135.

[14] Veres, S. M. and Luo, J. "A class of *bdi* agent architectures for autonomous control." *In Decision and Control*, 2004. CDC. 43rd *IEEE Conference* on, volume 5, pages 4746-4751, Dec. 2004.

[15] Mehta, N. et al. "Transfer in variable-reward hierarchical reinforcement learning." *Machine Learning*, 2008. 73(3): p. 289-312.

[16] Baxter, J. "Reinforcement Learning in POMDP's via Direct Gradient Ascent." *In Proc. 17th International Conf. on Machine Learning*, 2000: p. 41-48.

[17] Williams, R. "Simple statistical gradient-following algorithms for connectionist reinforcement learning." *Machine Learning*, 1992. vol. 8(2): pp. 229-256.

[18] Kozlowski, K.P. "Modeling and control of a 4-wheel skidsteering mobile robot." *International journal of applied mathematics and computer science*, 2004. 14(4): p. 477-496.

[19]Meier, L., Ross, D.W. and Glaser, M.B., "Evaluation of the Feasibility of Using Internal Redundancy to Detect and Isolate Onboard Control Data Instrumentation Failures," Tech. Report AFFDL-TR-70172, Wright-Patterson Air Force Base, Dayton, Ohio, Jan., 1971.

[20]Deyst, J.J. and Deckert, J.C., "Maximum Likelihood Failure Detection Techniques Applied to the Shuttle RCS Jets," *J. Spacecraft and Rockets*, AIAA, Vol. 13, 65-74, 1976.

[21]Patton, R.J., Frank, P.M., and Clark, R.N., *Fault Diagnosis in Dynamic Systems: Theory and Applications*, Prentice-Hall, Englewood Cliffs, NJ, 1989.

[22]Zhang, Y., Jiang, J., "Bibliographical review on reconfigurable fault-tolerant control systems," *Annual Reviews in Control*, Elsevier Ltd, Volume 32, Issue 2, December 2008, Pages 229-252.

[23]Williams, C**.**B**.,** and Nayak, P.P., "A Model-Based Approach to Reactive Self-Configuring Systems," in *Proceedings of AAAI-96,* pages 971-978, AAAI, AAAI Press, Cambridge, Mass., 1996.

[24]Zhang, Y.M. and Jiang, J, "Integrated Active Fault Tolerant Control Using IMM Aproach". *IEEE Transactions on Aerospace and Electronic Systems,* IEEE, Vol. 37, No. 4, 1221-1235, October 2001.

[25]Nasir, A. and Atkins, E.M. "Fault tolerance for Spacecraft Attitude Management," AIAA *Guidance, Navigation, and Control Conference*, Toronto, Ontario, Aug. 2-5, 2010 (AIAA-2010-8301).