Comprehensive Fault Tolerance and Science-Optimal Attitude Planning for Spacecraft Applications

by

Ali Nasir

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Aerospace Engineering)
in The University of Michigan
2012

Doctoral Committee:

Associate Professor Ella M. Atkins, Co-Chair
Professor Ilya V. Kolmanovsky, Co-Chair
Professor Jessy W. Grizzle
Professor Pierre T. Kabamba
Professor N. Harris McClamroch

To James William Fulbright and Richard Ernest Bellman

Acknowledgements

First of all, I would like to thank the Fulbright Program in Pakistan. Specifically, United States Education Foundation in Pakistan (USEFP), Higher Education Commission (HEC) of Pakistan, and the Institute of International Education (IIE) for their financial and administrative support. I would also like to thank Pakistan Space and Upper Atmosphere Research Commission (SUPARCO) for letting me avail this opportunity to persue higher studies. Then of course, I would like to thank my advisors Professor Ella M. Atkins and Professor Ilya V. Kolmanovsky for their enormous support and guidance through the course of my degree. It has been a true honor working with them. Both offer different flavors in terms of research and academic approach. I have had a great experience working with them academically and knowing them personally. I would also like to thank Professor Jessy W. Grizzle for guiding me through the initial phase when I was looking for a research advisor, helping me understand the non liniear control systems, and also being there at the end as one of my thesis committee members. Further, I would like to thank Professor N. Harris McClamroch for his guidance in various phases during my PhD and helping me understand the spacecraft dynamics and control. Last but not the least, I would like to acknowledge the guidance and education that I received from Professor Pierre T. Kabamba, Professor Semyon M. Meerkov, Professor Demos Teneketzis, and my colleagues Shinung Ching, Mike Hafner, Ryan Eubank, and Catherine Mcghan. Oh and how could I forget to mention my adorable wife Zahra for her support and sporadic feedback on my approach towards the academics and general life.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

There was a time when astronomers could only study space objects from Earth's surface, a time when communication between two people living on opposite sides of the Earth could take months or even years. Spacecraft now enable us to watch videos from Mars rovers and to almost instantly communicate worldwide. While space missions have vastly improved our exploration and communication capabilities, they have also introduced a need to manage new challenges that arise due to the harsh nature of the space environment. Hazards in the space environment include solar and galactic radiation [101][9][98][5], space debris and meteoroids [9], extreme temperature changes [101], and differential charging [83][42]. These hazards can damage spacecraft components or even subsystems causing failures and inability to complete the mission or interruptions in mission operation.

There are numerous examples where space missions suffered from failures due to the harsh space environment [9]. In 1994, Telsat, Canada's Anik E-1 communications satellite, suddenly began to spin out of control. Two hours later its sister satellite, Anik E-2, also without warning, began to spin out of control [54][9]. Telsat engineers quickly determined that the gyroscopic guidance system on both satellites had failed causing an interruption of cable TV, telephone, newswire, and data transfer services throughout

Canada. By activating a backup guidance system, engineers restored Anik E-1 to service in about eight hours. Anik E-2's backup system, however, failed to activate, leaving Telsat with the unpleasant prospect of losing a $228 million asset and revenues of an estimated $3 billion [9][1]. Telsat engineers restored Anik E-2 to service in August 1994 [50]. The Earth sensors were still operating normally [18], and the onboard attitude control computers were also fully operational, but not reprogrammable. Therefore, special Earth observation stations were constructed at each end of Canada to monitor the satellite's position, using its functional control jets to finely position the satellite.

The Far Ultraviolet Spectroscopic Explorer (FUSE) mission [65] suffered multiple reaction wheel and gyro failures [56][11][91]. Each time a new reaction wheel or set of gyroscopes was lost, a new control scheme was manually developed by engineers on the ground and uploaded to the spacecraft. Each new failure caused interruption in mission operation and loss of valuable mission time. In the end, FUSE kept operating with only one reaction wheel and magnetic torque bars [91]. Tracking and Data Relay Satellite (TDRS)-1 had 37 reported single event upsets during the major part of its solar activities [9][103]. The most serious incidents for the TDRS spacecraft were those related to the attitude control system processor electronics. Rapid manual intervention was required to prevent loss of control of the satellites. Several studies concluded that these anomalies were due to surface charging [41].

Due to the high cost associated with space missions, successful completion of mission goals and uninterrupted operation are highly desirable. One way to deal with most anomalous situations arising during the mission is to provide the spacecraft with a comprehensive plan for mission completion that includes post-failure recovery strategies.

This approach can increase the likelihood of uninterrupted mission operation through high-speed onboard reconfiguration as compared to the traditional approach of activating safe mode and waiting for the ground station to upload a new response for a faulty situation. Another advantage of a software-based reconfiguration approach is reduced cost compared to methods that involve carrying thus launching redundant hardware. Onboard planning for space missions has been achieved in the past [68][107][22][66][67][85]. There are also missions where some tolerance for faults has been incorporated [68][95]. Scientists have developed algorithms for onboard planning using Iterative Refinement Search [22][85] and similar methods based on constraint satisfaction [66][67][95] to build plans that can automatically adapt activity schedules to changes in mission goals or sensor measurements. While these methods are fast and reactive (i.e., suitable for implementing online), the resulting plans may not be optimal with respect to scientific data that is collected in the presence of uncertainties.

To deal with planning with autonomous fault management, additional capabilities of fault detection and reconfiguration are required. The combination of fault detection and reconfiguration provides fault tolerance. Researchers from the control systems and artificial intelligence communities have developed methods for achieving fault tolerance [111][104][68]. The fault tolerance methods developed by control system engineers are based on dynamics models [111], whereas the methods developed by engineers in the artificial intelligence community are based on compositional models [68]. Since a spacecraft is a system that has dynamics and compositional behaviors, *comprehensive fault tolerance* requires models of both. To the best of our knowledge, there is no planning framework for spacecraft to-date that implements comprehensive fault

tolerance. This fact motivated us to develop such a capability. Although the models formulated in this thesis are focused on optimal attitude maneuvers, our methods can be generalized.

This thesis introduces the Comprehensive Fault Tolerant Science-Optimal Attitude Planning (CFT-SOAP) framework. To implement this framework, we have used the theory of Markov Decision Processes (MDPs) also known as Stochastic Dynamic Programming (SDP) [15][84][58]. Both the MDP and SDP refer to a mathematical framework that relies on algorithms such as value iteration and policy iteration [89][58] for the purpose of generating optimal decisions under uncertainty for systems that exhibit the Markov property [84]. CFT-SOAP does not implement dynamics-based fault detection. Instead it connects with an external dynamics-based fault detector enabling CFT-SOAP to generate optimal mission plans with comprehensive fault tolerance.

1.2 Major Challenges

One of the most important challenges in developing CFT-SOAP is the integration of compositional-model fault tolerance with dynamics-model fault tolerance. Reasoning with compositional models is accomplished via a discrete-time finite state space governed by logical inference. Reasoning with dynamics models requires a continuous-time state space governed by differential equations. Rather than combining compositional and dynamics models we achieve comprehensive fault tolerance through information sharing. In this way, we keep the dynamics and compositional models in their original form. Also, we can use the well-developed reasoning methods based on dynamics and logic augmented by shared information. CFT-SOAP therefore must compute optimal mission-related actions under given fault and configuration information, draw

conclusions about the faults, resolve conflicts between the dynamics-based and logic-based fault decisions, and make optimal reconfiguration decisions.

Since CFT-SOAP is a comprehensive framework, it inherits the complexities of both compositional and dynamics models. For example, an attitude maneuvering plan must be optimal with respect to fuel/energy consumption and reward obtained from scientific data collected. Fault tolerance for attitude maneuvering requires knowledge of probabilities of false alarms and missed detections based on both logic and observer (dynamics) fault detection models. This integration requires numerous design parameters in the MDP which in turn introduces a high level of model complexity. Additionally, all models handled by an MDP are applicable only when obeying the Markov property. Per the Markov property, future state probabilities must depend only on the present state and any choice(s) made for the present state. System states therefore must be defined such that transitions to any future are determined from the present state and are independent of all past states.

Complex models are difficult to manage due to the curse of dimensionality associated with MDPs [80][79]. Although computing technology continues to advance, space-grade processors lag state-of-the-art. Therefore, models and algorithms must be developed to fit computations into the available resources if the MDP policies are to be computed online. In this thesis, we manage complexity by decomposition of the CFT-SOAP MDP into three separate MDPs for planning, fault detection, and reconfiguration thereby reducing the computational effort required to generate the policies and the memory space required to store them. We also present approximate dynamic programming (ADP) approaches for managing the decomposed MDPs as will be described below. Regardless of the

decomposition and the ADP approaches, it might be more feasible (as we suggest in this thesis) to compute a set of optimal policies offline and upload them to the spacecraft when required.

1.3 Technical Approach

Our approach is to use a multi agent architecture where goal-based task planning, fault detection, and reconfiguration are divided into separate MDP formulations that interact and share information with each other. As shown in Figure 1.1, our approach integrates four software agents [89]. Three of the agents on the left are based on MDPs whereas the fourth agent is a non-MDP-based interface agent. The interface agent interprets observations from the composition and dynamics-based control and configuration models in the form of states for the three MDP-based agents. The states for each of the MDP-based agents contain information from both the compositional and dynamics models. In response to the states provided by the interface agent, the MDP-based agents transmit the corresponding optimal policies. These policies are interpreted and distributed by the interface agent as commands for the control architectures within the dynamics and composition-based models. MDPs were selected because they support complex decision making in the presence of uncertainties.

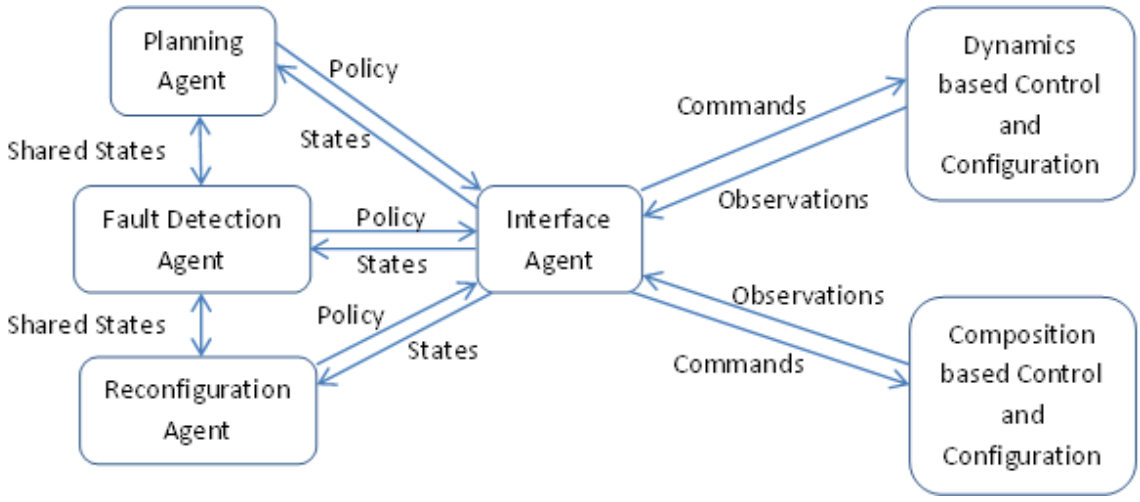

Figure 1.1: MDP based multi-agent approach

1.4 Original Contributions and Innovations

The contributions of this thesis are in formulations of MDPs and approximate dynamic programming approaches for spacecraft planning, fault detection, and reconfiguration. Innovations involve the application of techniques to share information between compositional and dynamics models, and in decomposing the problem to reduce complexity. Specific contributions and innovations are listed below.

1.4.1 Contributions

The following are the original contributions of the work presented in this thesis:

- An integrated Comprehensive Fault Tolerant Science-optimal Attitude Planning (CFT-SOAP) MDP has been formulated for spacecraft missions (Chapter 3). This MDP can produce optimal policies for actions related to planning, fault detection, and reconfiguration simultaneously. Computational complexity of the integrated MDP has been analyzed for a spacecraft attitude maneuvering case study. CFT-

SOAP is the first planning framework with integrated comprehensive fault tolerance.

- To reduce computational complexity, a decomposition strategy for the CFT-SOAP MDP based on interdependencies of states (Chapter 3) has been developed. As a result of the decomposition, the problem is split into manageable sub-problems, several of which are static.
- An MDP-based framework for science-optimal attitude planning with consideration of failure probability (Chapter 4) has been developed. Tradeoffs between selections of various design parameters have been explored through spacecraft maneuvering examples.
- An approximate dynamic programming (ADP) algorithm for science-optimal attitude planning with consideration of failure probability has been developed based on decomposition of science goals (Chapter 4). A case study for ADP shows near-optimal performance.
- An MDP-based comprehensive fault detection and diagnostic framework has been developed. The framework has the ability to integrate with external dynamics-based fault detectors to enable use of general detection algorithms (Chapter 5) including but not limited to MDPs.
- An approximate dynamic programming algorithm for comprehensive fault detection has been developed based on a task based decomposition of states (Chapter 5).
- A comprehensive MDP-based optimal fault reconfiguration framework has been developed for a spacecraft attitude maneuvering system (Chapter 6).

- A detailed case study based on the failures that occurred in the Far Ultraviolet Spectroscopic Explorer (FUSE) mission has been presented. In this case study, a CFT-SOAP solution to automatically reconfigure in the presence of faults relevant to FUSE is compared with an alternative planning framework known as ASPEN [85] and a fault tolerance framework known as Livingstone [104].

1.4.2 Innovations

The following are the innovations of this work that together enable comprehensive fault tolerance and science-optimal planning.

- We have incorporated science rewards, dynamics-dependent maneuver costs, and failure probability into a unified spacecraft mission planning algorithm. This enables task-level planners to incorporate the effects of physics-based control schemes.
- We have developed conflict resolution schemes for co-existing fault detectors. These schemes improve fault detection quality over the one which is possible to achieve with separate dynamics-based and compositional models due to the ability to incorporate shared information.
- We have developed an integrated value function that incorporates terms related to mission accomplishment with fault management value formulated in terms of safety. This formulation is distinct from traditional fault-configuration mapping where reconfiguration decisions only depend upon the nature and location of faults. Our framework enables the spacecraft to respond to the failures in a way that depends upon mission context as well as location and severity of faults. This ability is especially useful for space missions with objectives to record or observe

events that occur infrequently. In such missions, if the spacecraft fails to collect important data, the mission will fail.

- We have demonstrated the application of the CFT-SOAP on a case study of a real space mission, i.e. FUSE, where MDP decompositions and additional variables in the state space specific to the requirements of the FUSE mission have been included. This provides insight into how the work presented in this dissertation can be applied in practice.

1.5 Thesis Outline

The thesis is divided into seven chapters. In Chapter 2, we present pertinent background and discuss related literature. Chapter 3 presents the integrated MDP-based framework that forms the basis of CFT-SOAP. This framework encompasses planning, fault detection, and reconfiguration, all in one MDP. The computational complexity of this framework has been illustrated through a simple example, providing motivation for the ADP formulations that reduce computational overhead via decomposition of the MDP into multiple smaller MDPs. Chapter 4 presents our implementation of science optimal attitude planning with consideration of failure probabilities using an MDP formulation. Spacecraft case studies included in the chapter illustrate the tradeoffs in parameter selection for the planning MDP associated with discounting the future rewards. Case studies to show the comparison of the MDP-based trajectories under high-risk versus low-risk environment are also described. The approximate dynamic programming algorithm used to mitigate complexity is also described in Chapter 4 along with case study to show the performance of the ADP. Chapter 5 presents conflict resolution algorithms for fault detection and diagnosis. It also presents a collaborative fault

detection and diagnosis algorithm based on an MDP formulation. An approximate dynamic programming algorithm approach is also used to mitigate complexity in the Chapter 5 MDP formulation. Chapter 6 presents an MDP based comprehensive reconfiguration framework. Chapter 7 includes FUSE case study and the comparison of CFT-SOAP with an ASPEN-Livingstone framework. Chapter 8 presents conclusions and future work.

Chapter 2

Background

This chapter provides a review of technical literature relevant for this thesis. Section 2.1 presents an overview of planning and scheduling for space missions without considering fault tolerance. Section 2.2 presents a review of constraint satisfaction problems including techniques that are primary alternatives of our approach for solving fault tolerant attitude planning problem. Section 2.3 presents some references and methods in hybrid systems or switched control systems that are extensively used for fault tolerant control. Section 2.4 describes existing frameworks for fault tolerant attitude planning. In Section 2.5, spacecraft attitude kinematics and dynamics models are presented. A review of the literature on spacecraft attitude estimation and control is presented in Sections 2.6 and 2.7 respectively. Section 2.8 presents a review of Bayes nets that is used in this thesis to model interdependence of probabilities of failures for spacecraft components. A review of Markov Decision Processes is presented in Section 2.9 and the Approximate Dynamic Programming literature is discussed in Section 2.10. A brief introduction to some basic Astrodynamics concepts useful in understanding the planning formulations of this thesis is presented in Section 2.11.

2.1 Spacecraft Mission Planning and Scheduling

There are five main ingredients of the classical artificial intelligence (AI) planning problem. A finite set of discrete states, a set of state-dependent actions, the specification of desirable or goal states, the specification of an initial state, and a search method to determine an optimal sequence of actions (i.e. the solution) that leads from initial state(s) to goal state(s). For a given size of the state and action spaces, the computational complexity of finding the solution depends upon the search method used. All the planning methods are centered on the method of search that they use to find a solution. Typical real-time schedulers, on the other hand, see the world as a set of resources and a set of resource-consuming tasks, requiring up to a known worst-case utilization of each computational resource (e.g., processor or communication). Schedulers allocate resources to tasks, assigning each a start time and resource set that guarantee all deadlines are satisfied, making tradeoffs as needed to degrade best-effort tasks given resource constraints [24][77]. One of the basic algorithms used for scheduling is earliest-deadline-first scheduling [106][57] where tasks are placed in a priority queue and whenever a scheduling event occurs (task finishes, new task released, etc.) the queue is searched for the task closest to its deadline. Another basic algorithm for scheduling is rate-monotonic (RM) scheduling [7][109] where priority is given to the tasks with shortest period.

Autonomous spacecraft task planning and scheduling have been achieved for a limited set of science missions [22][43][67]. Algorithms such as iterative repair [22] have been selected due to their ability to adapt existing plans without prohibitive computational overhead. Iterative repair supports continuous modification and updating of a current working plan in light of changing operational context. Iterative repair adapts an existing

plan by using search-based algorithms such as backtracking [68]. This results in plan improvement but optimality in general is not guaranteed by iterative repair due to the local nature of search in iterative algorithms. Reference [22] discusses the use of iterative repair techniques to support a continuous planning process as is appropriate for autonomous spacecraft control. This allows the plan to incorporate execution feedback such as early or late completion of activities and over- or under-utilization of resources. Another reference on integrated planning and scheduling is [66] that presents a Heuristic Scheduling Testbed System (HSTS). HSTS is a representation and problem solving framework that provides an integrated view of planning and scheduling. HSTS emphasizes the decomposition of a domain into state variables evolving over continuous time. This allows the description and manipulation of resources more complex than are modeled in classical task scheduling. The inclusion of time and resource capacity into the description of causal justifications allows a fine-grain integration of planning and scheduling and a better adaptation to problem and domain structure. HSTS puts special emphasis on leaving in as much temporal flexibility as possible during the planning/scheduling process to generate better plans/schedules with less computation effort. Schedules developed in HSTS implicitly identify a set of legal system behaviors. This is an important distinction with respect to classical approaches which, instead, specify all aspects of a single, nominal system behavior. In [107], a multi-agent planning system (MAPS), which is used to produce applicable action sequence under complex constraints, is built for autonomous planning. The planning model is capable of describing simultaneous activities with continuous time. The model can also handle resource and temporal constraints. The architecture of MAPS includes planning agents

(PAs) and a planning manager agent (PMA). Each subsystem in the spacecraft can be considered an agent, with the agents combined to complete the given goals. PMA manages the mission planning system and functions as a communication medium between PAs. All PAs must register with PMA before they can have any interactions with the others in the system.

2.2 Constraint Satisfaction Problems

Constraint Satisfaction Problems (CSPs) represent a class of AI planning problems where states belong to specific domains of values and there are constraints over allowable combinations of values for subsets of state variables. CSPs can be solved with algorithms that take advantage of the specific state-space formulation. A constraint satisfaction problem (CSP) requires a value, selected from a given finite domain, to be assigned to each variable in the problem, so that all constraints relating the variables are satisfied. A sequence of actions is then selected that allow the plan to satisfy goals and which allow constraints, numerical and symbolic, to be satisfied. Many combinatorial problems in operational research, such as scheduling and timetabling, can be formulated as CSPs. In [45], the authors explore the number of tree search operations required to solve binary constraint satisfaction problems. They show analytically and experimentally that the two principles of first trying the places most likely to fail and remembering what has been done to avoid repeating the same mistake twice improve backtracking search performance. In [34], Dechter identifies classes of problems that lend themselves to easy solutions, and develops algorithms that solve these problems optimally. Other useful references on CSPs by the same author include [33][35]. Brailsford et al. describe CSPs and solution techniques in [17], and also show how various combinatorial optimization

problems are solved using a constraint satisfaction approach. Constraint satisfaction approaches are compared with well-known operational research (OR) techniques such as integer programming, branch and bound, and simulated annealing. The constraint Processing (CP) approach in [17] is unlikely to be competitive with the best local search methods, such as simulated annealing, tabu search and genetic algorithms, if it is used in a pure form, since large regions of the solution space are often unexplored. However, if ideas from local search are incorporated, such as randomization and restart procedures, then CP becomes a serious competitor to local search for obtaining approximate solutions.

2.3 Hybrid Systems

The hybrid systems of interest to this work contain two distinct types of components, subsystems with continuous dynamics and subsystems with discrete dynamics that interact with each other. Note that CFT-SOAP has fault-tolerant control as part of the architecture which qualifies as a hybrid system in some sense although we did not use the theory of hybrid systems, strictly speaking, for developing CFT-SOAP. Hybrid systems arise in varied contexts for applications in manufacturing, communication networks, autopilot design, automotive engine control, computer synchronization, and chemical processes, among others. Hybrid systems have a central role in embedded control systems that interact with the physical world. They also arise from the hierarchical organization of complex systems and from the interaction of discrete planning algorithms and continuous control algorithms in autonomous, intelligent systems. A survey on modeling and control of hybrid systems is presented in [60]. This survey highlights certain characteristics of hybrid systems. A simple three fluid-filled tank system is used to illustrate some

modeling approaches. Variations to this example are used to further explore hybrid model characteristics. An expository discussion is also presented in [60] on analysis and control techniques for hybrid systems. Work on model-predictive control of discrete-time hybrid systems is presented in [78]. The algorithm abstracts the behavior of the hybrid system by building a ‘‘tree of evolution.’’ The nodes of the tree represent the reachable states of a process, and the branches connect two nodes if a transition exists between the corresponding states. A cost-function value is associated with each node, and based on this value the exploration of the tree is driven. Other references on hybrid systems include [3][6]. In [6], a brief introduction to the theory and applications of hybrid systems is presented and an outline of the papers in the associated special issue is given. In [3], output feedback control of a class of stochastic hybrid systems is discussed.

2.4 Architectures for Fault Tolerance and Mission Planning

Researchers in the artificial intelligence (AI) community have proposed a variety of architectures for planning/scheduling and plan execution [68][115][107][89]. Most represent state as a list of symbolic (discrete) feature/value pairs, enabling search-based algorithms to decompose, select, and sequence activities appropriate for the designated task-level goal and the observed system state. For spacecraft for which operations involve nontrivial uncertainty, reasoning is typically based on Bayesian and/or Markov Decision Process (MDP) models [15]. The MDP builds optimal policies that allow an agent to act with incomplete or uncertain information about itself or its environment. Note that MDP solves slightly more general form of the AI planning problem where state transitions involve uncertainties and possess Markov property. Although common in the literature, few AI systems have successfully been deployed in space systems due to their

computationally-intensive deliberative and often difficult-to-validate nature. Rather than extensively trade the nontrivial set of AI architectures, we primarily reference Remote Agent [68][46][76] due to its emphasis on fault detection and reconfiguration and its focus on space applications. Below, we describe this architecture in more detail since, although 15 years old, it represents one of the most successful multi-layer AI architectures implemented and deployed on a spacecraft. Following our description of Remote Agent as a representative AI architecture focusing on fault management, we describe fault-tolerant control [74][111][13]. Fault-tolerant control represents more traditional guidance, navigation, and control (GNC) models in which physical continuous-valued state and control input vectors are related through physics-based models to describe the motion of a system through its environment. As described below, fault tolerance is then achieved through a mode-based supervisor and control law adaptation.

2.4.1 The Remote Agent

Researchers from the Jet Propulsion Laboratory (JPL) and NASA Ames developed the Remote Agent AI architecture to enable autonomous onboard mission management [68][46]. Remote Agent was tested on the Deep Space One (DS-1) spacecraft and consisted of five main components including: 1) Planning Experts (PE), 2) Mission Manager (MM), 3) Planner and Scheduler (PS), 4) Smart Executive (EXEC), and 5) Mode Identification and Reconfiguration (MIR). Planning Experts (PE) are on-board software modules that assist a task planner/scheduler either by computing solutions or by requesting new goals. For example, a navigation PE might request updates to main

engine thrust goals based on its determination of the spacecraft orbit, and the attitude PE might provide estimated duration of specified turns and resulting resource consumption.

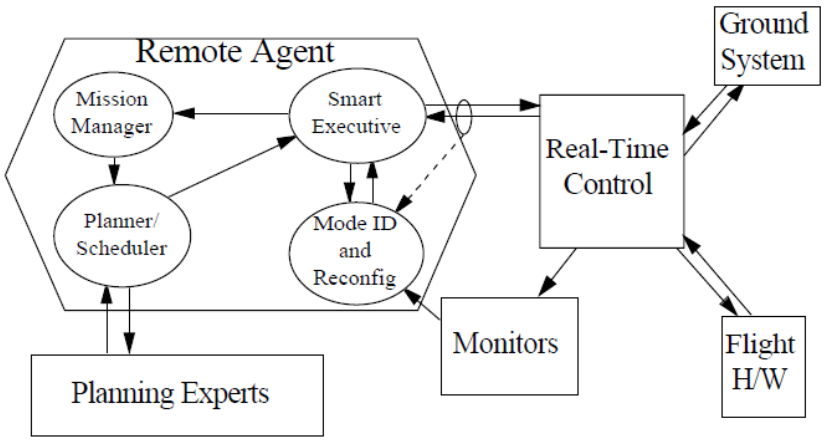

Figure 2.1: The original Remote Agent architecture [68]

The Mission Manager (MM) initiates planning/scheduling activities based on the long-term mission profile and execution status updates. The [smart] executive (EXEC) provides spacecraft status data and requests plans from the mission manager. The mission profile is provided at launch and can be updated from the ground. MM determines the goals to achieve during the next mission phase, and combines them with current spacecraft status. By adding constraints to the plan request, MM restricts PS to generate only plans that are coherent with the overall mission. This decomposition of planning into long-term mission planning and short-term task planning enables RA to undertake an extended mission with minimal human intervention. Such multi-resolution planning architectures have previously been used for applications such as telescope science scheduling.

The planner/scheduler performs Iterative Refinement Search (IRS) [67] and chronological backtracking to define a task set that extends the existing partial plan. A

plan database input to the Heuristic Scheduling Testbed System (HSTS) [115] described above records the consequences of each problem-solving step and performs consistency maintenance and propagation. Domain constraints are specified in the Domain Description Language (DDL) [32] within HSTS. Throughout, system state is described as a finite set of symbolic state variables with tokens used to describe both action and state literals. PS uses classical search-based planning and scheduling and is efficiently implemented with persistent parallel threads. PS is able to handle non-classical goal types such as periodic goals, accumulation goals, and default goals.

EXEC is a robust event-driven and multi-threaded plan execution system. It provides a framework in which specific mission goals and spacecraft state can be used to customize control, diagnosis, and reconfiguration capabilities autonomously. It can request and execute plans involving concurrent and interdependent activities potentially with uncertain timing and outcomes. EXEC decomposes planned tasks into primitive commands executed closed loop (i.e., as a function of state). This enables the planner to reason at a higher level of abstraction. EXEC's design also supports close integration between activity decomposition and fault response. EXEC is built on the Execution Support Language (ESL) [43] providing parallel execution, synchronization, error handling, and property locks [32]. EXEC loads and executes each plan while monitoring task execution and spacecraft status through Mode Identification (MI). When a plan is completed successfully, EXEC provides current status to MM and asks for a new plan. If task execution fails, EXEC puts the spacecraft into safe mode but autonomously asks MM rather than a ground operator for an alternate plan (unless MM can no longer resolve the problem in which case ground operators must be involved). EXEC achieves

robustness by exploiting flexibility to create and modify plans based on goals and observations and by handling execution failures using deductive plan repair (Mode Reconfiguration (MR)).

Livingstone [104] provided the Mode Identification and Reconfiguration (MIR) functionality of Remote Agent. Livingstone is a discrete model-based controller inserted between high-level feed-forward reasoning and low level feedback control layers in a physical system. MIR proposes activities to migrate a system (spacecraft) to a configuration that achieves a configuration goal. It has a sensing component, Mode Identification (MI), and a commanding component, Mode Reconfiguration (MR). Its model is declarative, compositional, and stochastic with concurrency support. Mode Identification tracks changes in spacecraft status using input from EXEC as well as a spacecraft system model. It predicts state values and compares them with monitored values. In case of discrepancy, it predicts the malfunction or fault most likely to explain the discrepancy. Mode Reconfiguration (MR) assists EXEC in generating recovery procedures. On the occurrence of a fault, EXEC invokes MR with current fault information from MI. EXEC also provides MR with global constraints and goals. MR performs deduction and search in a reactive loop using fast propositional reasoning through unit propagation along with conflict directed best-first search. Figure 2.2 shows the architecture of Livingstone.

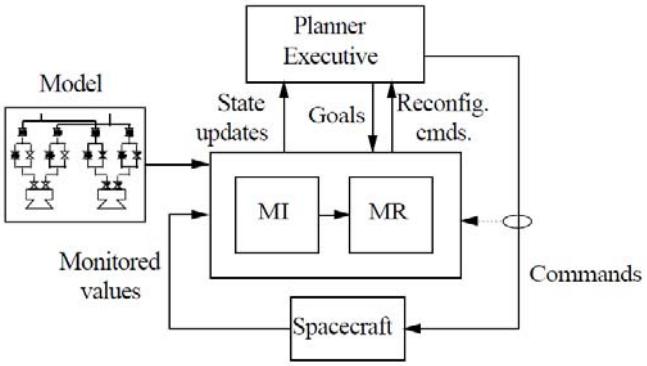

Figure 2.2: Livingstone architecture [68]

2.4.2 Fault Tolerant Control Systems

While the MIR capability of RA can identify and respond to faults via discrete event state (mode) models, MIR does not itself regulate continuous force/torque commands, nor does it adapt to properties of physics-based models except by switching between pre-specified modes. The control systems community has studied fault management primarily in the context of adapting physical models and control commands. A class of architectures known as Fault Tolerant Control System (FTCS) has emerged. A typical FTCS has three layers [114][74][13]. The lowest layer is a reconfigurable feedback control law with state estimator. The middle layer is a fault detection and diagnosis (FDD) [111] scheme. At the top level is a supervisor that manages reconfiguration of the FDD and control layers. Figure 2.3 shows an example fault tolerant control system architecture. In this figure, **x** is the state vector for dynamic system, **u** is the control input vector, **y** is the output vector, **F** is the vector of fault flags, and *M* is a scalar indicating configuration mode of the reconfigurable controller.

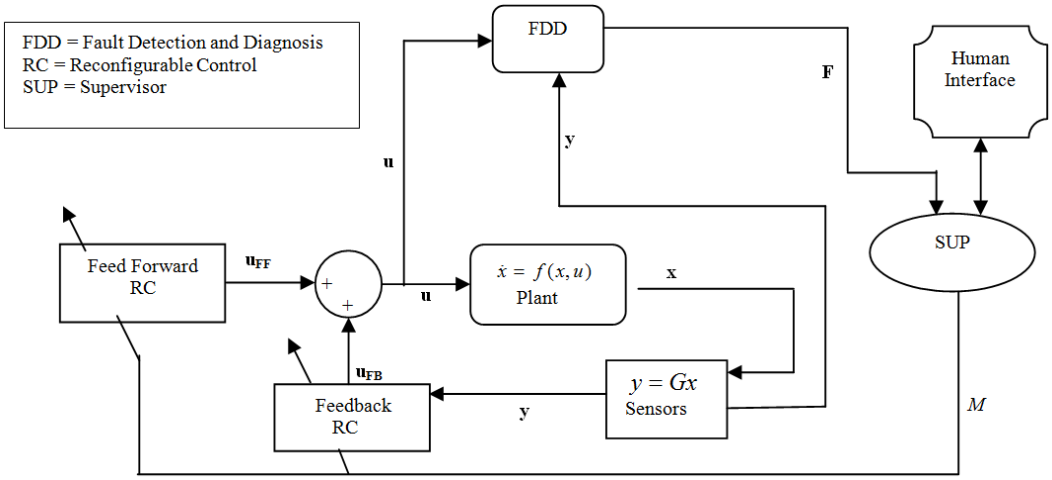

Figure 2.3: Fault tolerant control architecture

FTCS have adopted numerous control law formulations. In this manuscript we focus on a scheme applicable to spacecraft attitude control. Two types of fault tolerant controllers exist: passive strategies (robust control) [74] and active strategies (controllers for which reconfiguration is based on projection or on-line controller adaptation). Passive fault-tolerant control for a spacecraft uses a robust controller, providing a baseline upon which an active fault tolerant scheme could be implemented. Since the robustness of a controller has an effect on fault detection efficiency, a tradeoff between the two must be established. An active control approach can be implemented using projection (i.e. controller selection from predesigned alternatives) or adaptive feedback control approaches (i.e. online controller redesign) [13]. The main purpose of this FTCS layer is to adapt to anomalous situations and either to restore nominal performance, if possible, or to gracefully degrade [113].

Fault Detection and Diagnosis (FDD) predicts faults (**F**) from residual signals [21] that indicate the deviation of actual behavior (**x**) of the dynamics from the nominal behavior (f(**x**, **u**)) based on sensor measurements (**y**) and fault effect models. When the system is

fault-free, all residuals should be driven to zero by the controller. An FDD scheme could be based on state or parameter estimation or a mixture of both. FDD can be classified into two groups: model-based approaches and data-based approaches. A fault detection and diagnosis scheme should be robust, especially when model-based [112]. With sound but imperfect FDD models or incoming data, missed detections or a false alarm can occur. The decision-making in FDD can be made robust using methods such as statistical data processing, averaging, fuzzy decision-making, and adaptive thresholds [112][48]. Another issue is to distinguish between disturbances (unwanted forces/torques exerted by the environment), noise (distortion in sensor output signal), and faults (malfunctioning of sensors or actuators). Disturbance decoupling methods can also be applied. For a complex system such as a spacecraft, decoupling of residuals from a set of integrated disturbances sometimes makes the residual completely or partially insensitive to some faults. From the point of view of a spacecraft, an FDD scheme should be able to detect multiple simultaneous faults, both abrupt and incipient.

Supervision is the top FTCS layer and is responsible for reconfiguration decisions (*M*) based on information from FDD and its own reasoning algorithms. Supervision schemes have been developed to manage diagnostic information and on-line controller restructuring. Supervision modules have been implemented with methods [13][74] including Failure Mode Effect Analysis (FMEA) [13], Intelligent Computing, Fuzzy Logic, Neuro-Computing, Genetic algorithms, and Probabilistic reasoning. FMEA models the effect of faults on observable system parameters by providing data on how each fault impacts these parameters. The supervisor can be implemented using extended

state machine or parallel state machine logic with transition probabilities based on knowledge of the system.

2.4.3 Representational Gaps

Remote Agent models a spacecraft as a concurrent transition system with multiple components and operating modes, while FTCS models the spacecraft as a rigid or flexible body with associated kinematics and dynamics. Both rely on sensor data fused into state estimates and translated to control output, where "control" is defined for Remote Agent as general action primitives and for a FTCS as a vector of physical servo/motor commands. RA reasons about spacecraft components and their interactions but typically does not manipulate physical dynamics/kinematics parameters. On the other hand, FTCS can reason on the basis of equations of motion but is unable to reason about component interactions and task-level algorithm or software implementation properties. These differences define *representational gaps* in both architectures. One might be tempted to bridge these gaps by extending the capabilities of either RA or FTCS. Incorporating dynamics and kinematics reasoning in symbolic models is possible but difficult due to the tradeoff in [discrete model] resolution versus search-space tractability. On the other hand, incorporating qualitative component and interconnection details in FTCS not only requires a state-based supervision architecture that can reason about system-wide interactions but also the ability to reconfigure (re-plan) based on component failures and events associated with components such as communication channels, processing elements, the payload, etc. This thesis therefore proposes the use of both classes of fault management algorithms with an appropriate set of interfaces to facilitate conflict resolution.

2.5 Spacecraft Modeling: Kinematics and Dynamics

There are multiple ways of representing spacecraft attitude kinematics [93]. Equations (2.3.1a), (2.3.1b), and (2.3.1c) represent alternate ways to model the kinematics of the spacecraft as a rigid body.

$$\dot{q} = \frac{1}{2}\left(q_4\Omega - [\Omega \times q]\right)$$
$$\dot{q}_4 = -\frac{1}{2}\left(\Omega^T q\right) \tag{2.3.1a}$$

$$\dot{\mathrm{R}} = \begin{bmatrix} 0 & \omega_3 & -\omega_2 \\ -\omega_3 & 0 & \omega_1 \\ \omega_2 & -\omega_1 & 0 \end{bmatrix} \mathrm{R} \tag{2.3.1b}$$

$$\begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\theta}_3 \end{bmatrix} = \begin{bmatrix} \cos\theta_2 & \sin\theta_1 \sin\theta_2 & \cos\theta_1 \sin\theta_2 \\ 0 & \cos\theta_1 \cos\theta_2 & -\cos\theta_2 \sin\theta_1 \\ 0 & \sin\theta_1 & \cos\theta_1 \end{bmatrix} \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix}. \tag{2.3.1c}$$

In (2.3.1a), q is a 3 × 1 vector of the first three elements of spacecraft attitude quaternion with respect to an inertial reference frame; q_4 is a scalar representing the fourth element of the quaternion. Ω is a 3 × 1 vector of spacecraft angular velocities in a body-fixed frame. In (2.3.1b), R is the 3 × 3 rotation matrix for the spacecraft whereas ω_1, ω_2, and ω_3 represent components of Ω. In (2.3.1c), $[\omega_1 \quad \omega_2 \quad \omega_3]^T$ represent angular velocities in the body-fixed frame and $[\theta_1 \quad \theta_2 \quad \theta_3]^T$ represent Euler angles roll, pitch, and yaw, respectively, with respect to the inertial frame. There are other ways to represent the attitude, e.g. Euler-Rodriguez parameters [93] [51], but other forms are not used in this thesis. The quaternion representation is common for spacecraft as it has no singularities and requires only four continuous-valued quantities in its representation. Euler angles do

have singularities, while rotation matrices have no singularities but must be represented with nine values.

The attitude dynamics of a rigid spacecraft can be represented as

$$\begin{aligned}\dot{\omega}_1 &= \frac{1}{J_1}\left[\left(J_2 - J_3\right)\omega_2\omega_3 + u_1\right],\\ \dot{\omega}_2 &= \frac{1}{J_2}\left[\left(J_3 - J_1\right)\omega_1\omega_3 + u_2\right],\\ \dot{\omega}_3 &= \frac{1}{J_3}\left[\left(J_1 - J_2\right)\omega_2\omega_1 + u_3\right].\end{aligned} \tag{2.3.1d}$$

In (2.3.1d), (J_1, J_2, J_3) are diagonal components of 3 × 3 inertia matrix in a body-fixed frame (we assume the inertia matrix to be diagonal) and (u_1, u_2, u_3) represent control inputs. There are a number of ways to control spacecraft attitude [102] even with two control inputs instead of three [55]. In [55] a discontinuous feedback control strategy has been constructed which stabilizes the spacecraft to any equilibrium attitude in finite time. The results of the paper show that although standard nonlinear control techniques do not apply, it is possible to construct a stabilizing control law by performing a sequence of maneuvers. Also, for attitude determination, a number of ways have been developed to estimate the attitude from the sensor readings which may or may not provide accurate measurements, e.g. [61] where Kalman filtering has been used to estimate the attitude with gyroscopes that have both drift and bias errors.

2.6 Spacecraft Attitude Estimation

Robust spacecraft attitude estimation is required for most missions and requires fault tolerance. Most dynamics-based fault detectors are based on output estimation. One of the most useful resources in the literature of spacecraft attitude estimation is a survey

paper by Lefferts et al [61]. This work presents a summary of experience in the Kalman filtering of spacecraft attitude and offers two possible implementations of the Kalman filter for systems with attitude sensors and gyros with noise terms describable by a first-order Markov process. The difference in the two schemes is only in the choice of frame for the update, for example using the complete four-component quaternion versus using the truncated quaternion where one component has been eliminated. Crassidis and Markley [25] present a minimum model error approach for attitude estimation. The approach is developed for three-axis stabilized spacecraft. Based on the implementation example included in [25], their algorithm is shown to be robust and accurate, able to estimate attitude with or without gyro measurements. The functional form of the optimal estimation approach involves a gradient search and a linearization technique with a linear Riccati transformation. This algorithm is shown to be computationally efficient and accurate for generating state estimates based on an implementation example. Results using this algorithm indicate that an MME-based approach accurately estimates the attitude of an actual spacecraft with the use of only magnetometer sensor measurements. Crassidis and Markley have also published their work on an unscented Kalman filter for spacecraft attitude estimation [27] and attitude estimation using modified Rodriguez parameters [26]. Both these authors along with Cheng have published a survey on modern attitude estimation methods [28]. This survey presents a quaternion estimation filter (QUEST), extended QUEST and the backwards-smoothing extended Kalman filter. Filters that propagate and update a discrete set of sigma points rather than using linearized equations for the mean and covariance are also reviewed. A two-step approach is discussed with a first-step state that linearizes the measurement model and an iterative

second step to recover the desired attitude states. These approaches are all based on the Gaussian assumption that the probability density function is adequately specified by its mean and covariance. Other approaches that do not require this assumption are also reviewed, including particle filters and a Bayesian filter based on a non-Gaussian, finite-parameter probability density function on SO(3). Finally, the predictive filter, nonlinear observers and adaptive approaches are shown.

2.7 Spacecraft Attitude Control

Attitude control is fundamental for ensuring spacecraft stability thus requires fault tolerance. Multiple attitude control schemes are often required for implementing full reconfigurable control. A careful presentation of spacecraft attitude control methods is given by Bong Wie [102]. This book treats the basics of dynamic systems modeling and control. The attitude control and stabilization problems of rigid spacecraft under the influence of reaction jet firings, internal energy dissipation, or momentum transfer via reaction wheels or control moment gyros (CMGs) are discussed in this book. These techniques can provide a good support for our proposed framework especially when different types of redundant actuators are used in the spacecraft for fault tolerance. A variety of control problems of spinning as well as three-axis stabilized spacecraft are also treated in [102]. Emphasis is placed on large-angle reorientation maneuvers in which a spacecraft is required to maneuver about an inertially fixed axis as fast as possible, but within the saturation limits of rate gyros and reaction wheels. Such maneuvers are fundamental for science data collection missions where targets of interest can only be observed by slewing the spacecraft through a sequence of large-magnitude motions. The attitude control and momentum management problem of a large space vehicle in low

Earth orbit such as the International Space Station is also treated in the book. Advanced spacecraft control problems of developing CMG steering logic and optimal jet selection logic are also treated.

There has also been some interesting work done on magnetic control of spacecraft attitude in [29][62][82][94][110]. For example, in [29], a formulation for reconfiguring the control based on magnetic dipole moment modulation for the attitude control of Earth-pointing satellite is presented. Spacecraft control with two torques has also been extensively studied e.g. in [55]. Spacecraft control with two-torques is very useful for implementing fault tolerance.

2.8 Spacecraft Modeling with Bayes Networks

The Bayes network [89] is a way of representing dependence relations between random variables and is used for efficient computation of joint and conditional probabilities. We use Bayes nets for modeling the internal composition of a spacecraft by realizing that failure of any component is a random event and failures of components within the spacecraft depend upon each other in a way that can be determined from the interconnection and interaction of components with one another. Hence, a Bayes net can be constructed to succinctly represent conditional probability tables (CPTs). This thesis studies the use of Bayes nets to intuitively represent CPTs associated with spacecraft fault diagnosis. For example, consider a one degree of freedom (1DOF) reaction wheel system where a battery supplies power to two electronics boards (one of which is redundant for fault tolerance) that can drive the reaction wheel. A simple Bayes net model for failure probabilities of this system is shown in Figure 2.4.

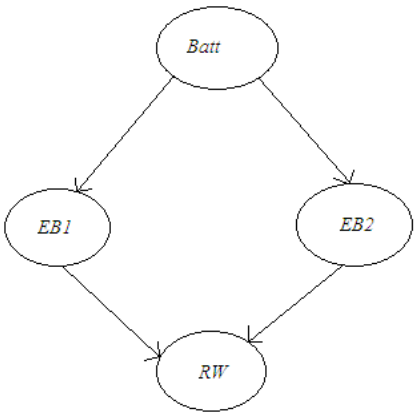

Figure 2.4: Bayes net example

From above figure, notice that the failure of the battery (Batt) affects the failure of the electronics boards (EB1 and EB2), and the failure of the boards affects the failure of the reaction wheel (RW). Furthermore, failure of the reaction wheel is conditionally independent of the failure of battery given definitive knowledge of whether the electronics boards have failed. This type of model can be used to solve for the probabilities of failure of any components or subsystems given failure information about any other component(s) or even when no information is given. There are quite a few methods for deriving probabilities from Bayes nets [89] including enumeration, variable elimination, and local propagation. The computational and memory requirements for some of the methods are shown in Table 2-1. In this table, *n* is the number of nodes in the Bayes Net, and all nodes are assumed to be binary, e.g. fail/not-fail. Also note that the local propagation method has the lowest computational complexity but it is only applicable on Bayes nets that have a poly tree structure (i.e. no cycles or multiple paths connecting one node to another).

Table 2-1: Computational complexity of methods for solving Bayes Nets [89]

Method	Applicability	Memory Requirement	Computational Cost
Enumeration	general	O(n)	O($n2^n$)
Variable Elimination	general	O(2^n)	O(2^n)
Local Propagation	polytrees	O(n)	O(n)
Clustering	general	O(2^n)	O(2^n)
Conditioning	general	O(n)	O(2^n)

2.9 Markov Decision Processes (MDPs)

An MDP is a controlled Markov chain [58] that is solved using a discrete stochastic dynamic programming (SDP) algorithm, e.g. value iteration or policy iteration [84][89]. Value iterations are applied to the optimal control problem to maximize an expected discounted reward function of the form

$$V^{Pol}(s) = E\left[\sum_{t=0}^{\infty} \gamma^t R(s^t, \mu^t) \mid Pol, s^0 = s\right]. \tag{2.3.3-1}$$

Here, s^t represents state after t actions, and μ^t is the action applied in state s^t according to a policy Pol (s^t is a random variable). 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 the future rewards have lower value. We assume that R is bounded from above and below. The policy that selects the optimal action may be found as

$$Pol^*(s_i) \in \arg\max_k \left(R(\mu_k, s_i) + \gamma \sum_{j \in S} T(s_j \mid \mu_k, s_i) V(s_j) \right). \tag{2.3.3-2}$$

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 [89]:

$$V_{t+1}(s_i) = R(s_i) + \max_k \left(\sum_{j \in S} \gamma T(s_j \mid \mu_k, s_i) V_t(s_j) \right) \tag{2.3.3-3}$$

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. $T(s_j \mid \mu_k, s_i)$ is the probability of transitioning from state s_i to s_j by executing action μ_k.

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

$$Itr = \left[\log\left(\frac{2R_{\max}}{(1-\gamma)\varepsilon} \right) / \log\left(\frac{1}{\gamma} \right) \right]. \tag{2.3.3-4}$$

Here ε is the required tolerance of the solution satisfying

$$\left\| V_{t+1}(\varsigma_i) - V(\varsigma_i) \right\| < \varepsilon, \forall i. \tag{2.3.3-5}$$

The inequality (2.3.3-5) is ensured by [89]

$$\left\| V_{t+1}(\varsigma_i) - V_t(\varsigma_i) \right\| < \varepsilon \left(\frac{1-\gamma}{\gamma} \right). \tag{2.3.3-6}$$

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. As described in [58], Equation (2.3.3-3) converges to a unique solution. The solution of Equation (2.3.3-3) achieves its maximum value of the right hand side in Equation (2.3.3-1). If the policy is

calculated using (2.3.3-2) with solution of (2.3.3-3), it will be optimal with respect to (2.3.3-1).

The value iteration algorithm used to solve (2.3.3-3) and find an optimal policy from (2.3.3-2) is shown in Figure 2.5.

Step 0. Initialization:
Set $V^0(s) = 0$ for all $s \in S$
Fix a tolerance parameter $\varepsilon > 0$
Set $t = 1$.

Step 1. For each $s \in S$ compute:

$$V_t(s) = R(s) + \max_{\mu \in M} \left\{ -C(\mu) + \gamma \sum_{s' \in S} T(s, \mu, s') V_{t-1}(s') \right\}$$

Step 2. If:

$$\| V_t - V_{t-1} \| < \varepsilon (1 - \gamma) / 2\gamma,$$

calculate:

$$P^*(s) = \max_{\mu} \left\{ -C(\mu) + \gamma \sum_{s' \in S} T(s, \mu, s') V_t(s') \right\}$$

else, set $t = t + 1$ and go to Step 1.

Figure 2.5: Value Iteration algorithm

2.10 Approximate Dynamic Programming

Approximate Dynamic Programming (ADP) can be used to reduce the computational complexity of MDPs. There are three important books dedicated to this topic, each representing different communities. Bertsekas and Tsitsiklis [10] provide a primarily theoretical treatment of the field using the language of control theory. This text [10] uses neural network approximations to overcome the "curse of dimensionality" and the "curse of modeling" that have been bottlenecks to the practical application of dynamic

programming and stochastic control to complex problems. This methodology allows systems to learn about their behavior through simulation, and to improve their performance through iterative reinforcement [learning]. Sutton and Barto [97] describe the field from the perspective of artificial intelligence/computer science [97], starting with intuitive examples and a definition of reinforcement learning. They then present three fundamental approaches to reinforcement learning: Dynamic Programming, Monte Carlo, and Temporal Difference methods. Subsequent chapters build on these methods to generalize to a spectrum of solutions and algorithms. Powell [79] uses the language of operations research, with more emphasis on the high-dimensional problems that typically characterize the problems in this community. In [53], the authors present an algorithm that dynamically performs hierarchical decomposition of factored MDPs. Their algorithm is based on determination of causal relationship between states. Communication-based decomposition methods for decentralized MDPs are presented in [44]. A goal-based decomposition approach (similar to the approach adopted in this thesis) is presented in [16]. In [16], the decomposition is based on the additive terms in the reward function that correspond to different sub-goals. Decomposed MDPs are assigned sub-goals based on decomposition of the reward function. Optimal policies are computed for each sub-goal and finally merged together using a value function heuristic and best-first search to generate an approximate policy for the original task.

2.11 Astrodynamics

Since this thesis deals with spacecraft missions, fundamentals of two-body orbital motion play a key role when discussing the data collection windows for science observation targets. There are six major parameters of a spacecraft orbit, as described below. .

2.11.1 Orbital Elements

The six classical orbital elements [8] are shown in Figure 2.5:

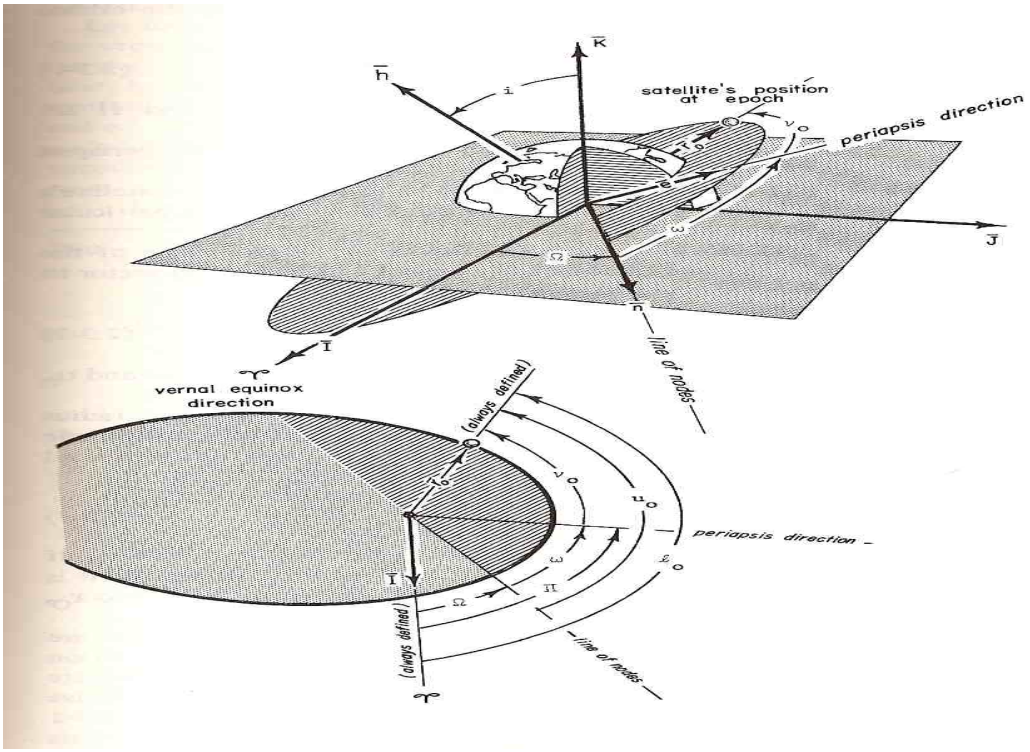

Figure 2.6: Orbital elements [8]

1. *Semi major axis (a):* A constant defining the size of the conic orbit.

2. *Eccentricity (e):* A constant defining the shape of the conic orbit.

3. *Inclination (i):* The angle between the **K** unit vector and the angular momentum vector (Figure 2.6).

4. *Longitude of the ascending node (Ω):* The angle in the fundamental plane, between the **I** unit vector and the point where the satellite crosses through the fundamental plane in the northward direction (ascending node) measured counterclockwise when viewed from the north side of the fundamental plane.

5. *Argument of periapsis (ω):* The angle in the plane of the satellite's orbit, between the ascending node and the periapsis point, measured in the direction of the satellite's motion.

6. *True Anomaly (v):* Angle between the line joining the satellite's center of mass with the center of mass of the central body and the line of periapsis measured in the direction of motion of the satellite.

In ideal two-body motion, five of the six orbital elements remain constant, while the sixth, true anomaly ν, precesses over a range from 0 to 2π radians as the spacecraft revolves around the central body. We exploit the periodic nature of true anomaly in our MDP formulations to enable a cyclic and finite state-space despite a potentially infinite time horizon.

Chapter 3

An Integrated Markov Decision Process Modeling Framework

We present a Markov Decision Process (MDP) based framework to automatically compute optimal strategies for integrated Comprehensive Fault Tolerant Science-Optimal Attitude Planning (CFT-SOAP). Because CFT-SOAP combines a broad suite of compositional models into a single framework, deliberation in CFT-SOAP is computationally expensive. This chapter presents an integrated spacecraft decision-making model for CFT-SOAP to introduce the baseline capability. This chapter then investigates decomposition of the full decision process into multiple decision-making units, each with lower computational complexity, that together provide the capabilities in the full integrated architecture but for which conflicts in decisions must be carefully resolved.

The goal of the full CFT-SOAP architecture is to provide an integrated goal-based planning solution that also incorporates fault detection, conflict resolution, and comprehensive reconfiguration in one framework. In addition to describing the decision-making modules, we assess the interdependence of various components of the state feature vector. Based on the interdependences, we propose a decomposition approach which is based on task reformulation. As a result we obtain three separate MDPs: one for planning, one for fault detection with conflict resolution, and one for comprehensive

reconfiguration. Throughout, we assume the spacecraft always follows a stable drift orbit such that attitude control is the only activity requiring physical actuation. We also assume the policies can either be developed offline and uploaded or else that sufficient onboard resources are available. Although we introduce a means to decompose CFT-SOAP into the three modules listed above, we focus on model development in the MDP, not on real-time policy execution within a specific spacecraft onboard computing environment. While our cost functions may penalize energy consumption for actions such as pointing or communicating (see Chapter 4), we typically also assume the spacecraft has sufficient stored energy to execute planned action sequences, although a simple energy model is included for the FUSE spacecraft case study in Chapter 7.

3.1 Problem Formulation

3.1.1 Goals

The major goals of CFT-SOAP are to:

- Calculate optimal attitude maneuvers and data collection actions that maximize science reward and minimize the probability of possible failures.
- Detect failures in a manner that minimizes probabilities of missed detection and false alarms using fault information from an observer-based dynamic fault detector and conflict resolution actions.
- Reconfigure spacecraft components and control laws to maximize expected performance with respect to resource consumption and probability of successful mission completion.

3.1.2 Problem Statement

We seek to develop an MDP-based framework for computing optimal policies for fault tolerant attitude management of a spacecraft. We assume that the spacecraft is on a mission to collect scientific data from fixed targets. Because the spacecraft follows a fixed two-body orbit (with negligible drift due to perturbations), we assume the targets are visible to the spacecraft data collection sensors for pre-specified windows of visibility, and that these windows repeat as a function of orbit true anomaly ν. Note that periodic observation windows are not always available, particularly when Earth-observing spacecraft are not synchronized with Earth's rotation period, but for spacecraft observing distant celestial bodies this is a reasonable approximation.

Per Figure 1.1 in Chapter 1, we propose an MDP framework for autonomous onboard decision-making that contains planning, fault detection, and reconfiguration agents, as shown in Figure 3.1 below.

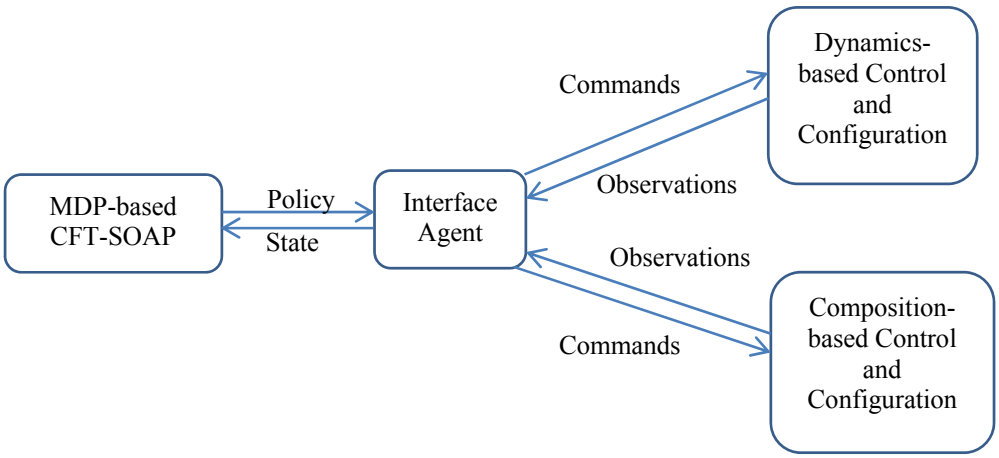

Figure 3.1: MDP-based integrated CFT-SOAP implementation.

3.2 MDP Formulation

The Markov Decision Process (MDP) supports decision-making under uncertainty. Optimal sequences of actions form a policy that allows an autonomous system to react appropriately to a wide range of observed situations, even those not on a "nominal" [deterministic or most likely] execution path. The MDP as used in CFT-SOAP is formally defined in this section.

3.2.1 States

MDP states for the science-optimal fault tolerant planner CFT-SOAP can be represented as follows:

$$
\begin{aligned}
&S = \{s_1, s_2, ..., s_N\} \\
&where, \\
&s_i = \{A_i B_i, z, v, Bl_i, Bo_i, U_i, O_i, V_i, sw_i, c_i\} \\
&A_i = \{a_i^1, a_i^2, ..., a_i^{n1}\} \\
&B_i = \{b_i^1, b_i^2, ..., b_i^{n2}\} \\
&Bl_i = \{bl_i^1, bl_i^2, ..., bl_i^{m1}\} \\
&Bo_i = \{bo_i^1, bo_i^2, ..., bo_i^{m2}\} \\
&U_i = \{u_i^1, u_i^2, ..., u_i^{m3}\} \\
&O_i = \{o_i^1, o_i^2, ..., o_i^{m4}\} \\
&V_i = \{v_i^1, v_i^2, ..., v_i^{m5}\} \\
&N \approx 2^{n1+n2+m1+m2} \times (n2+1) \times d \times d1^{m3} \times d2^{m4} \times d3^{m5} \times d4 \times d5
\end{aligned}
\tag{3.2.1}
$$

As shown above, each state consists of eleven types of information, including:

1. *Flags for Active Mission Related Actions A_i:* There are $n1$ binary flags indicating which of the mission-related actions are active (in progress) for the given state. In

our formulation, mission related actions include attitude maneuvering, data collection, and no-operation (NOOP) actions.

2. *Data collection flags* B_i*:* There are $n2$ binary flags indicating whether or not science data have been collected from the corresponding science target.

3. *Attitude pointing* z_i*:* This is a scalar having $n2+1$ values presuming one value for each of the $n2$ science targets and one additional value for pointing towards the visible ground station (or data relay satellite). Note that we do not consider all possible attitude pointings since in that case the state-space will become infinite. Also, we abstract above the specifics of pointing to the visible communication node, although this pointing angle would need to be tracked in real-time.

4. *True Anomaly* v_i*:* The true anomaly of the spacecraft is assigned d discrete partitions encompassing the 0 to 360 degree range. Again, a tradeoff between precision of the framework and size of the state-space is present.

5. *Logic-based fault flags* Bl_i*:* There are $m1$ binary flags indicating presence or absence of various faults based on observations and knowledge about the configuration and composition of the spacecraft. Examples of faults that may be included in Bl_i are failure of electronics, switches, valves, sensors, actuators etc.

6. *Observer-based fault flags* Bo_i*:* There are $m2$ binary flags indicating the presence or absence of modeled faults based on real-time observations and a priori knowledge about the dynamics of the spacecraft. Note that the sets of faults corresponding to the flags in Bl_i and Bo_i could be overlapping.

7. *Diagnostic Actions* U_i*:* There are $m3$ actions that can be used to collect additional data from the spacecraft. These data might be expensive to collect under normal circumstances thus not built into the default periodic schedule of operations. Each of these actions can be active or inactive ($d1$ in Equation (3.2.1) = 2).

8. *Diagnostic Observations* O_i*:* These represent $m4$ processed observations that indicate status of various components of the spacecraft. Note that processed observations are not sensor outputs, rather these are logic based inferences based on control commands, sensor outputs, and logical clauses that represent the compositional model of spacecraft components and subsystems.

9. *The threshold vector* V_i*:* This vector contains $m5$ thresholds that effect the fault flags Bo_i generated by an observer-based fault detection scheme. Here we assume that each threshold can affect one and only one fault flag in Bo_i whereas each flag in Bo_i can be affected by more than one threshold.

10. *Switching configuration* sw_i*:* This is a discrete-valued variable that represents the index of the current compositional configuration of the spacecraft. We assume there are $d4$ possible switching configurations.

11. *Control law configuration* c_i*:* This is a discrete-valued variable that represents the index of the currently-active attitude control law of the spacecraft. We assume that there are $d5$ possible attitude control law configurations. We do not model an orbital maneuvering capability in this thesis.

3.2.2 Actions

The following expressions represent the set of actions available in CFT-SOAP:

$$M = \left\{ \begin{matrix} \mu_{b0}, \mu_{b1}, ..., \mu_{bn2}, \mu_{bl1}, ..., \mu_{blm1}, \mu_{u1}, ..., \mu_{um3}, \mu_{v1+}, \mu_{v1-}, ..., \mu_{vm5+} \\ , \mu_{vm5-}, \mu_{sw1}, ..., \mu_{swd4}, \mu_{c1}, ..., \mu_{cd5}, NOOP_{\Delta v1}, ..., NOOP_{\Delta vn3} \end{matrix} \right\}$$

OR

$$M = \{M_B, M_{Bl}, M_U, M_V, M_{SW}, M_C, M_{NOOP}\}$$

where

$$\begin{aligned} M_B &= \left\{\mu_b^0, \mu_b^1, ..., \mu_b^{n2}\right\} \\ M_{Bl} &= \left\{\mu_{bl}^1, \mu_{bl}^2, ..., \mu_b^{m1}\right\} \\ M_U &= \left\{\mu_u^1, \mu_u^2, ..., \mu_u^{m3}\right\} \\ M_V &= \left\{\mu_v^{1+}, \mu_v^{1-}, \mu_v^{2+}, \mu_v^{2-}, ..., \mu_v^{m5+}, \mu_v^{m5-}\right\} \\ M_{SW} &= \left\{\mu_{sw}^1, \mu_{sw}^2, ..., \mu_{sw}^{d4}\right\} \\ M_C &= \left\{\mu_c^1, \mu_c^2, ..., \mu_c^{d5}\right\} \\ M_{NOOP} &= \left\{\mu_{\Delta v}^1, \mu_{\Delta v}^2, ..., \mu_{\Delta v}^d\right\} \end{aligned} \tag{3.2.2}$$

There are seven types of actions. First is the attitude maneuvering/data collection action set M_B. For actions of this type, we define a set of true anomaly windows for each of $n2$ targets in which the targets are visible to data collecting equipment:

$$W = \left\{\left[v_l^1, v_u^1\right], \left[v_l^2, v_u^2\right], ..., \left[v_l^{n2}, v_u^{n2}\right]\right\} \tag{3.2.3}$$

Here, v_l^k is the lower limit of the window of visibility of target k, while, v_u^k is the upper limit of the window of visibility of target k. Besides window of visibility, every target has another attribute that indicates whether the data from the target is to be collected only once in the mission or once per orbit around the central body:

$$p = \{p_1, p_2, ..., p_{n2}\} : p_i \in \{0,1\} \tag{3.2.4}$$

Here, $n2$ is the number of targets and p_k is a binary variable indicating whether target k requires periodic data acquisition (p_k = 1) or not (p_k = 0). An action μ_k in M_B could be either a data collection action or an attitude maneuver action. Data can only be collected

if the spacecraft is pointed toward and is within the window of visibility of the target from which data is to be collected.

Next are logic-based fault flag switching actions M_{Bl}. These actions can switch the fault flags for various components of the spacecraft *on* or *off*. As such they are "virtual actions" that set the internal state of the CFT-SOAP diagnosis engine.

Diagnostic input actions M_U are also available. These actions initiate diagnostic data collection activities. Note that the actions of type M_U set the diagnostic actions to 1. These actions are aperiodic; they become dormant automatically once executed, until an event or decision reactivates them.

Next are the threshold variation actions (or conflict resolution actions) M_V. These actions are used to vary the thresholds of the observer-based fault detector to tune fault decisions that conflict with the logic-based fault decisions. Each of these actions increases or decreases a false alarm/missed detection threshold by a fixed amount.

The fifth action type is the compositional configuration switching action M_{SW} used to change the current switching configuration of the spacecraft that may effectively stop the usage of faulty components and/or bring alternate healthy components into use. Also included are control law reconfiguration actions, M_C, used to change the law governing attitude control based on health status of the actuators and sensors involved.

Finally, the seventh type of actions is the no-operation action M_{NOOP}. This action results in the spacecraft doing nothing for any particular window (change) in true anomaly.

3.2.3 Reward Function

The reward function can be defined as

$$R(s(A,B,z,v,Bl,Bo,U,O,V,sw,c)) = R_1(B,z,v,Bl,Bo) + R_2(Bl,Bo,U,O,V) + R_3(A,B,Bl,Bo,sw,c) \quad (3.2.5)$$

We define the reward function as a sum of three functions. We now discuss what should be reflected in each reward functions for the spacecraft domain.

The first term R_1 in the above reward function represents science mission rewards and penalties and incorporates:

- Reward of collecting data from a science target. This type of reward is useful for emphasizing important targets versus less important ones.
- Reward of pointing towards a science target from which data has not yet been collected. This type of reward is useful for motivating the spacecraft to point towards a target in advance, if possible, rather than pointing towards the target just-in-time.
- Reward for completing a science mission. This type of reward motivates the spacecraft to complete the mission rather than just collecting as much data as possible with minimum risk.
- Penalty of being in a state which has no path of positive probability to any of the goal states. Goal states are the states in which all the scientific data have been collected.

A specific equation for R_1 is provided in Section 4.3 (Equation (4.3.1)).

The second term in the reward function represents fault-detection-related rewards and cost penalties. It should incorporate:

- Penalty on conflict(s) in fault detection between logic-based and observer-based fault flags. This term motivates the two fault detection schemes to consider each other's fault calls before (in case of logic-based fault detector) or after (in case of observer-based fault detector) making their own fault calls.
- Reward of correct detection based on probabilities of false alarms and missed detections. Correct detection reward will trade off with the penalty on any conflict(s) in fault detection to encourage false fault calls to be resolved or avoided.
- Reward of information available to the fault detection algorithm. This term motivates execution of information gathering actions that are helpful in obtaining useful information about the health status of the spacecraft especially in situations when there is persistent conflict between fault detectors. Examples of information gathering actions include short-term firing of thrusters to monitor the resulting accelerations, and switching from a primary sensor to a backup sensor to reconcile differences in readings, etc.

It is important note that in order to avoid negative rewards resulting from the penalties, all the penalties are defined as negative exponentials throughout this thesis. A specific equation for R_2 is provided in Section 5.9 (Equation (5.9.1)).

The third term in the reward function represents rewards related to fault-based reconfiguration including:

- Penalty on being in the states where an ongoing mission-related action is inapplicable or undesirable. This term demotivates reconfigurations that may cause problems for an on-going mission-related action. The reconfiguration strategy is encouraged to opt for other choice(s) if possible.
- Penalty on the states which have no path of positive probability connecting them with one of the goal states (goal states are states in which all mission objectives are achieved). This term motivates the reconfiguration strategy to keep spacecraft in a position to complete the mission even if other choices of reconfiguration are safer.
- Penalty on the unsafe or sub-optimal configuration of the spacecraft hardware and/or control law for a given fault and mission state. This term will trade off with the above term that motivates mission completion in situations where the safest possible reconfiguration action results in a state that has no path of positive probability connecting it with one of the goal states.

A specific equation for R_3 is provided in Section 6.2.2 (Equation (6.2.1)).

3.2.4 Action Costs

We assume that the costs associated with actions of type M_{Bl}, M_V, M_{SW}, M_C and M_{NOOP} are zero as these are simple computational or placeholder (NOOP) actions that do not require the spacecraft to consume extra energy to sense or apply physical forces or torques. The cost associated with actions of type M_B and M_U can be defined such that the energy, time, and fuel consumption required to complete these actions is penalized. A specific equation for the cost related to M_B is provided in Section 4.3 (Equation (4.3.7)).

3.2.5 Transition Probabilities

Transitions between MDP states are of two types. The first type of transitions models the effects of applied actions. For example, if we apply a data collection action, the action-dependent transitions reflect that either data are collected successfully or they are not collected. The second type of transitions model underlying uncertainties in different parameters of the states. For example, if we apply a data collection action, a reaction wheel or gyroscope can fail that does not affect data collection but results in a state that was not intended or dictated by the applied action. For some actions such as those involving flipping a switch or a flag, it can be assumed that nothing happens during the execution of actions i.e. none of the state variables change other than those intended to change by the action that is executed. For actions that are not instantaneous e.g. mission related actions, all possible transitions have to be modeled. This in general is a very difficult task. Therefore, it is desirable to use approximations. For example, instead of modeling probabilities of all possible failures that can happen during an action, one can model the probability that at least one failure will occur.

3.2.6 Bayes Net Structure (Compositional Model)

The MDP defined above requires an underlying Bayes Net where all the probabilistic dependencies between switches, fault flags (both logic-based and observer-based), control law configurations, diagnostic observations, diagnostic inputs, and threshold values are represented. The initial conditional probability tables must be at least approximated to form the basis from which the joint distribution can be computed in real-time from the more compact Bayes net representation. For finite horizon cases such as

what we experience with a day-to-day spacecraft activity planner, we can use a dynamic Bayes net [31] where conditional probabilities can depend upon time.

3.2.7 Dynamics Model

An underlying model of dynamics corresponding to each fault configuration is also presumed to be known a priori. This assumption is reasonable for consideration of the faults related to the attitude control sensors and actuators. This allows the MDP to switch between existing control law configurations rather than adapting numerical control law parameters in real-time. A spacecraft has a finite set of possible failure situations, each modeled in our state vector, thus capturing a suite of controllers to handle them all is realistic.

3.3 Closed loop execution of the MDP policy

A flow chart of the CFT-SOAP MDP formulation and uplink is shown in Figure 3.2, whereas closed loop onboard execution for the MDP is shown in Figure 3.3. Note that in Figure 3.2, the need for the new policy is determined based on the system-level failure or a combination of faults that can not be handled by the fault tolerance capabilities of the current MDP. This is different from today's execution sequences that do not have the ability to reconfigure in the presence of faults. In case of CFT-SOAP, the original MDP policy has a number of reconfiguration options built-in. Figure 3.2 depicts the unlikely case where none of the reconfiguration actions built into the executing MDP policy are applicable.

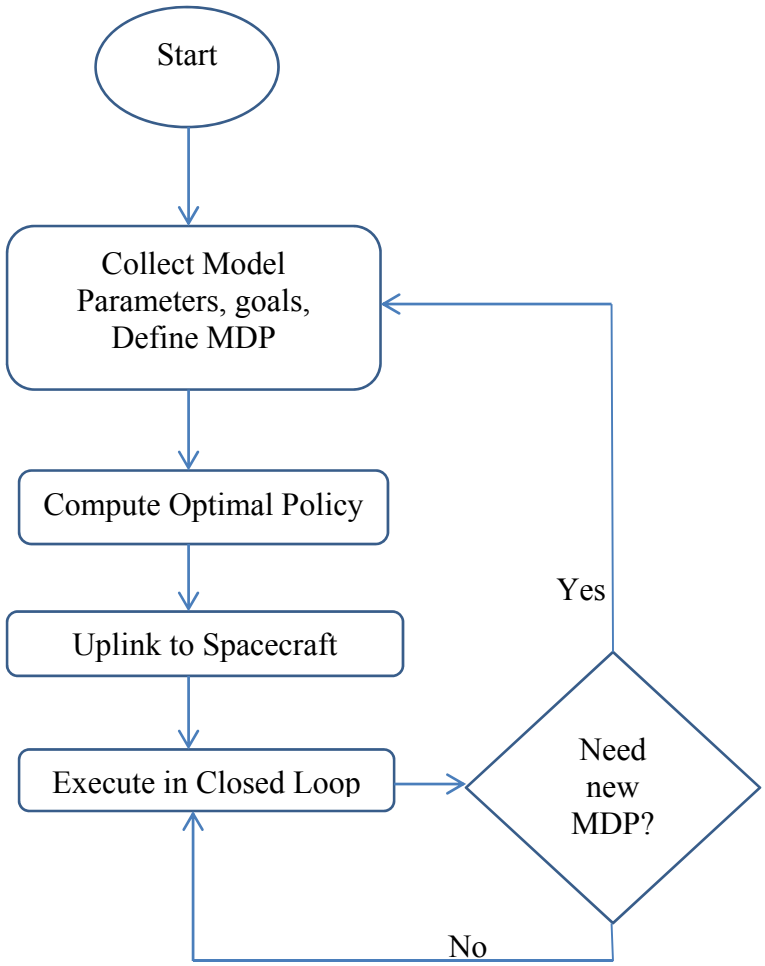

Figure 3.2: MDP modeling, uplink, and execution flowchart

In Figure 3.3, OBFDD stands for Observer Based Fault Detection and Diagnosis. y denotes the output vector from the dynamics, estimated in real-time by an inertial measurement unit (gyroscopes, accelerometers, and magnetometers) supplemented by star trackers, etc. to improve attitude computation accuracy. Y_l is the output from current, voltage, temperature, and pressure sensors. The real-time executive gathers information from system-wide modules and forms a state vector S. This state vector is sent to the MDP policy module which returns the corresponding optimal action. This action is interpreted as a full set of specific system level activities by the executive and these activities are then performed in a closed loop manner.

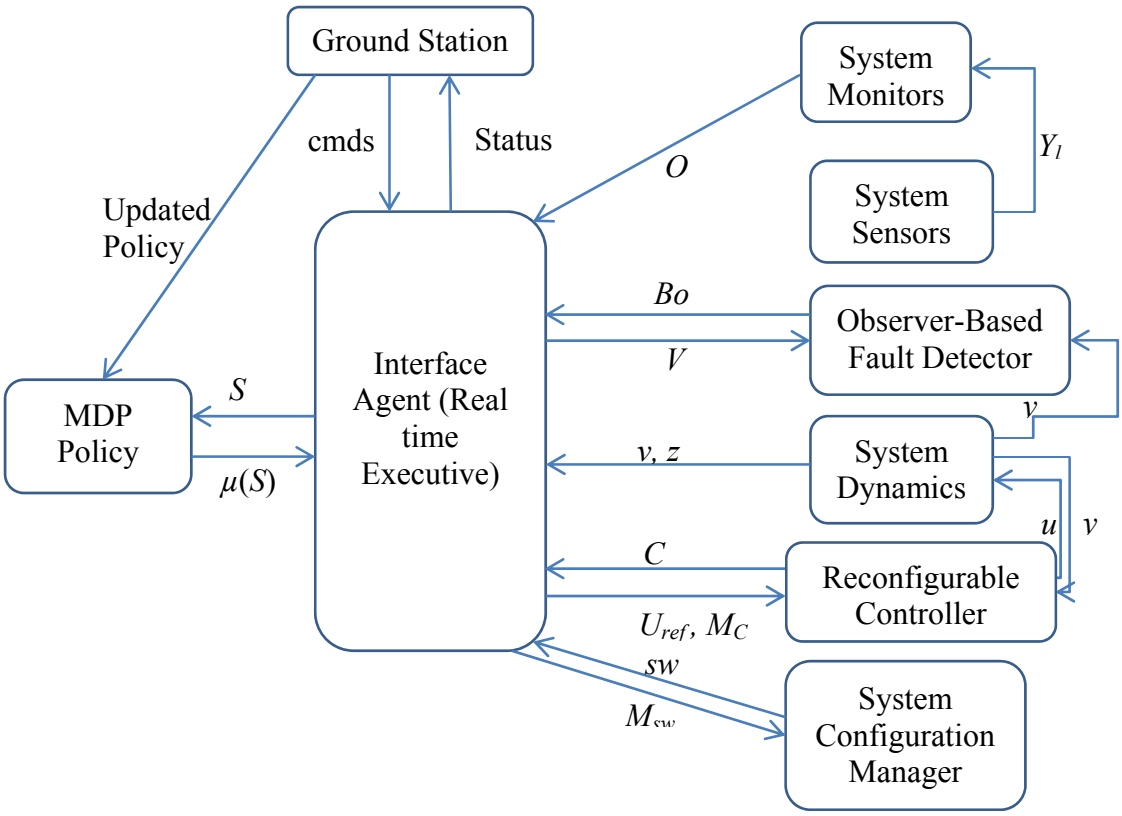

Figure 3.3: Real time closed loop execution of MDP

Figure 3.4 shows a detailed diagram of the operation of the executive. In Figure 3.4, the diamond-shaped nodes represent decisions. M_{DATA} and M_{ATT} are context-dependent representations of M_B indicating attitude maneuvering and data collection actions, respectively. Once the executive receives the next optimal action from the MDP policy module, it recognizes the action as related to fault detection (M_{FDD}), related to the spacecraft mission (M_B), or related to reconfiguration (M_{RC}). If an action $\mu(S)$ is recognized as M_{FDD}, it is further identified as either a threshold-changing action (M_V) or a diagnostic action (M_U), or a logic based fault flag switching action (M_{Bl}). In each case, the action is carried out by the appropriate modules. If the action $\mu(S)$ is recognized as

M_B, it is further identified as either an attitude maneuvering action (M_{ATT}) or a data collection action (M_{DATA}). In either case, the identified action is executed in a closed-loop manner. If the action $\mu(S)$ is recognized as a member of set M_{RC}, it is further identified as either a switching reconfiguration action (M_{SW}) or a control law reconfiguration action (M_C). In either case, the identified action is executed appropriately.

For sending a new state to the MDP policy and receiving a new action to execute, there is a condition X in the oval-shaped block in Figure 3.4. Condition X can be defined in multiple ways depending upon preferences of the designer. An example of a condition or test X is *"Return true if/when the previous action has been completed or the values in Bo or O have changed since the last time state information was processed*."

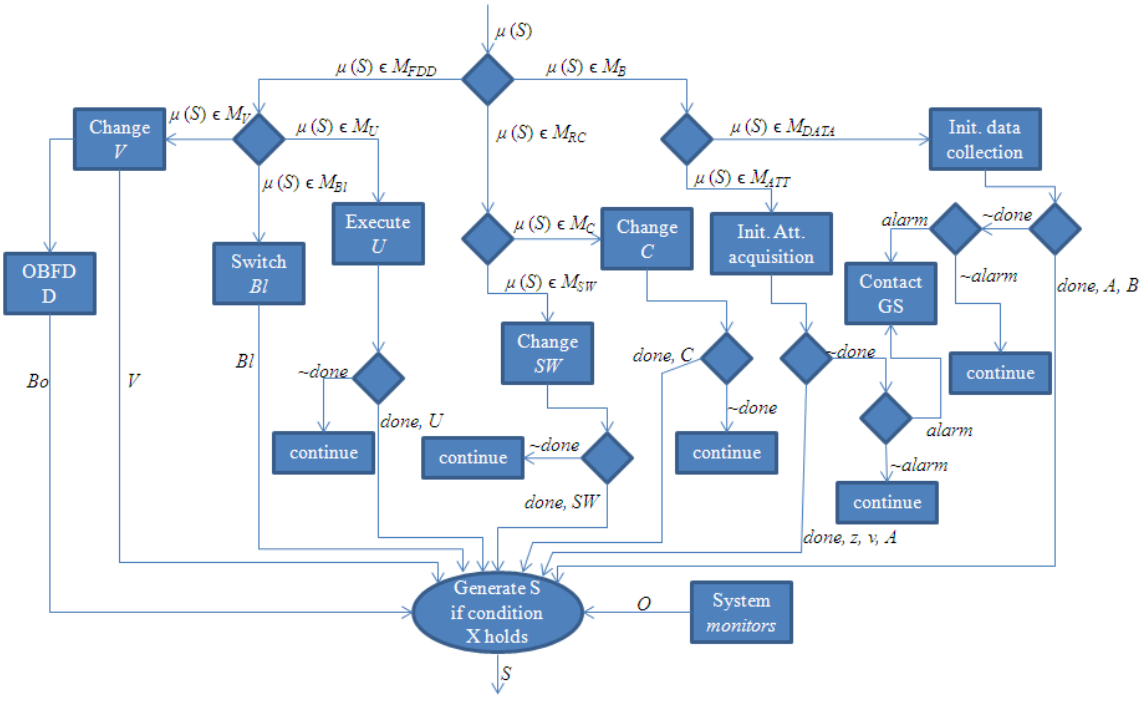

Figure 3.4: Detailed diagram of the MDP executive

3.4 Interdependencies and State Decomposition

This section studies model interdependencies and possible decomposition of various components of the MDP formulation in Section 3.2. There are some state parameters that are required explicitly by planning or fault detection or reconfiguration whereas some parameters are required by more than one of these activities. Figure 3.5 illustrates this concept. The following sections (3.4.1-3.4.3) propose a set of decompositions for planning, fault detection, and reconfiguration based on Figure 3.5 that each are more computationally tractable than the fully-integrated CFT-SOAP model but that together provide an approximation of the fully-integrated optimal CFT-SOAP solution.

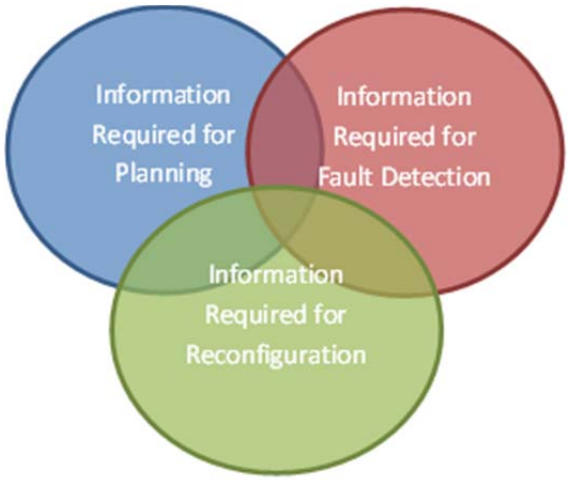

Figure 3.5: Concept of MDP decomposition

3.4.1 Planning

Recall that there are eleven major parameters in each MDP state (Section 3.2.1). Among these parameters, some are exclusively related to fault detection and reconfiguration. Examples include the set of diagnostic actions U, vector of processed observations O, and vector of thresholds V. Planning depends upon the following state parameters:

$$
\begin{aligned}
&S = \{s_1, s_2, ..., s_N\} \\
&where, \\
&s_i = \{B_i, z, v, Bl_i, Bo_i, sw_i, c_i\}.
\end{aligned}
\tag{3.4.1-1}
$$

Several parameters have been excluded in (3.4.1-1) to promote simplicity thus lowered computational complexity. Perhaps most debatable is the elimination of A_i that indicates status of the ongoing mission-related actions in state s_i. The exclusion of this vector is justified when actions are executed in a sequential order. In this way, an action from a plan is executed only when no other action is being executed. Other information that is excluded from (3.4.1-1) is needed purely for fault detection and diagnosis purposes. We can further simplify the state space for planning by generating separate plans for each switching and control configuration and each fault flag configuration. This will result in the following state formulation:

$$
\begin{aligned}
&S = \{s_1, s_2, ..., s_N\} \\
&where, \\
&s_i = \{B_i, z, v\}.
\end{aligned}
\tag{3.4.1-2}
$$

The set of actions related to planning are M_B and M_{NOOP}. Rewards and transition probabilities can be extracted from descriptions above in Section 3.2 (for specific details, see Sections 4.3.3 and 4.3.4).

3.4.2 Fault Detection

Fault detection requires information about fault flags, processed measurements, diagnostic actions, and ongoing mission-related actions since these actions can affect the probabilities of failures. Due to characteristics of the space environment (e.g., magnetic field, ability to communicate with other spacecraft or ground-based radios), fault

probabilities may also depend upon the position of spacecraft in its orbit and its pointing. Therefore, we also need information of true anomaly and current pointing of the spacecraft for the fault detection MDP. The resulting state space used for fault detection is:

$$\begin{aligned} &S = \{s_1, s_2, ..., s_N\} \\ &where, \\ &s_i = \{A_i, v, z, Bl_i, Bo_i, U_i, O_i, V_i\}. \end{aligned} \tag{3.4.2-1}$$

The only state information excluded in (3.4.2-1) is spacecraft hardware configuration, control law configuration, and status of mission objectives. In some cases, true anomaly and attitude pointing can also be excluded by using failure probabilities with worst case values. Also, vector A_i can be excluded using the same strategy. This will yield the following simplified states:

$$\begin{aligned} &S = \{s_1, s_2, ..., s_N\} \\ &where, \\ &s_i = \{Bl_i, Bo_i, U_i, O_i, V_i\}. \end{aligned} \tag{3.4.2-2}$$

Actions related to fault detection are M_{Bl}, M_U, and M_V. Also included is an instantaneous no-operation action M_{NOOP} to represent an option of not changing any detection flag, threshold, etc. Rewards and transition probabilities can be extracted from Section 3.2 (for specific details, see Sections 5.9.3 and 5.9.4). Note that, to avoid executing inappropriate goal-seeking actions, output of this fault detection agent should be used in selecting the appropriate planning policy.

3.4.3 Reconfiguration

For making reconfiguration decisions given error or fault conditions, information is required about current logic (switch) and control configuration, current faults, ongoing mission-related actions, and mission status. Therefore, the states for reconfiguration may include:

$$S = \{s_1, s_2, ..., s_N\}$$
$$where,$$
$$s_i = \{A_i B_i, Bl_i, Bo_i, O_i, sw_i, c_i\}. \quad (3.4.3\text{-}1)$$

Here again, information related specifically to fault detection and diagnosis has been removed. The actions involved in reconfiguration are M_{SW} and M_C. Introduction of an instantaneous no-operation action M_{NOOP} is again useful. Transition probabilities and rewards can be extracted from the discussion in Section 3.2 (for specific details, see Sections 6.2.2). To avoid inappropriate reconfiguration actions, output from fault detection and planning should be used to update the state vector in (3.4.3-1). Also, the latest configuration and control law selection should be applied along with latest fault information to select the appropriate planning policy.

3.5 Computational Complexity and Real time Implementation

The computational complexity of MDPs grows exponentially with the size of the state space. In particular, if the value iteration algorithm is used to solve an MDP, the complexity is of the order $O(N^2M)$ where N is the size of the state space and M is the size of the action space. To provide intuition for complexity of the proposed formulation in Section 3.2, assume that there are 2 mission related actions, 3 mission related tasks, 4 spacecraft pointing orientations, 50 values of true anomaly (each possible separated by

approximately 7 degrees given true anomaly range 0-360 degrees), 5 logic-based fault flags, 3 observer-based fault flags, 2 diagnostic actions, 5 processed observations with *fail/normal* type (binary) values, 2 switching configurations, 2 control configurations, and 2 observer-based thresholds each with 5 possible values. Despite the fact that most features have few values, the resulting size of this example state space is

$$\begin{aligned} N &= 2^2 \times 3^2 \times 4 \times 50 \times 2^5 \times 2^3 \times 2^2 \times 2^5 \times 2 \times 2 \times 5^2 \\ \Rightarrow N &= 6,039,797,760,000 \end{aligned} \tag{3.5.1}$$

Equation (3.5.1) shows 6 tera-states and the resulting complexity (assuming only 10 actions) will become of the order 10^{25}. For a comprehensively-modeled space mission, the complexity could easily increase to 10^{50} or 10^{75}. Since our computational capabilities are still of the order of 10^9 instructions per second, the integrated MDP could take years to compute. We therefore need to either build faster computers or reduce the computational complexity of the MDP, a primary objective of this thesis. The first step of reducing complexity has been presented in Section 3.4. Subsequent chapters represent how MDPs related to planning and fault detection can be further decomposed by using additional approximations.

3.6 Chapter Summary

We have presented an MDP-based integrated formulation (CFT-SOAP) for implementing spacecraft goal-based mission planning, fault detection, and fault reconfiguration that takes into account both logic-based compositional and continuous-valued models. We have also analyzed the complexity of a simple integrated model demonstrating the need for abstractions and decompositions to simplify the MDP so that it can be executed on spacecraft.

Chapter 4

Science-optimal Spacecraft Attitude Planning with consideration of Failure Probabilities

In this chapter, we present a framework to generate an optimal sequence of actions for spacecraft missions. Our generated sequences are optimal in a sense that the expected reward of science data collected in the presence of the possible failures is maximized. To deal with issues such as changing reward functions and/or transition probabilities, we show how to implement our approach using a receding horizon optimization formulation. We also provide an algorithm for approximate dynamic programming (ADP) that can be used to reduce algorithm computational complexity to tractable levels with the possibility of increased solution cost. Examples are also included to illustrate the implementation of the framework.

4.1 Motivation

Spacecraft are used in a variety of missions involving data collection from one or more celestial objects or from locations on the Earth. An onboard capability to autonomously plan and re-plan spacecraft missions to maximize data collection, conserve on-board energy and fuel, and account for potential or actual failures can greatly increase the autonomy and value of spacecraft missions. This capability can also reduce requirements for human intervention, which increases mission operations cost and introduces delays to the mission, following anomaly or failure events.

Related work on autonomous planning of spacecraft missions includes [68] where iterative refinement search combined with simple heuristics is used to generate sequences of actions for complex missions. These algorithms are capable of generating plans that involve concurrent actions, tight constraints, limited resources, and close deadlines. However, most of this work does not include models of spacecraft attitude or orbital motion, except for treating pre-defined set of constraints over observation windows. Other frameworks include the SPIKE scheduler for Hubble Space Telescope [52] that uses both rule-based and neural network approaches to schedule science observations while satisfying observation window, instrument, and onboard resource constraints. The Jet Propulsion Lab (JPL) has developed two architectures, ASPEN [85] and CASPER [22], for use in a variety of spacecraft missions requiring planning, scheduling, and iterative repair of the executing plan.

In this chapter, we focus on modeling maneuvers in the context of symbolic task-level planning as a first step toward integrating science and motion-centric planning processes. Specifically, we present a planner and associated models for optimal sequential decision-making in the presence of uncertainty, where actions can either change the attitude of the spacecraft or collect science data from a celestial object.

To build optimal plans for spacecraft missions, the planner must minimize energy use and maximize scientific rewards in the presence of failures. In this chapter, we present a framework based on an MDP [15], [89] adapted to spacecraft mission planning. We apply this framework to a specific class of missions which involve spacecraft pointing at specific targets. The system state reflects the current pointing direction, current true anomaly, and the status of target visit states (visited/not visited). Actions direct the

spacecraft through the set of pointing states; we presume a target will be viewed if the spacecraft is pointed at that target. Probabilities of successfully executing an action or alternatively entering a failure mode are encoded in an MDP transition probability matrix. Immediate rewards of states depend on the number of targets from which the scientific data has been collected in that state and on attitude pointing for that state. We also incorporate costs of attitude maneuvering actions in terms of the slew magnitude involved and costs of data collection actions in terms of energy consumed.

Given the computational complexity of the MDP, optimal policy generation is best to conduct offboard, with the output being a comprehensive policy which is then executed onboard the spacecraft. Note that this off-board computation and upload of the optimal policy requires onboard memory space on the order of the number of states in the MDP, unless more efficient tests are developed, e.g., through construction of a decision tree [89] that matches abbreviated feature tests to policy actions. Also there is cost associated with uploading of the policy. However, the MDP policy offers a level of robustness through fault detection and reconfigurability that can offset these additional complexity costs. To potentially mitigate the costs associated with computing the MDP, receding horizon policies (e.g., [72]) can be computed more quickly. In a receding horizon formulation the value function of the generated MDP policy is used as a terminal cost.

For a potential onboard implementation of the MDP, a finite horizon approach may also be used, where policies are generated for a short time horizon and assume a finite set of scenarios. This approach is useful when the reward function and the transition probabilities change with time. To deal with the computational complexity, an approximate dynamic programming approach to compute near-optimal policies is

applied. In this chapter, we propose an approximate dynamic programming approach for decomposition of the MDP based on mission objectives.

4.2 Problem Formulation

Assumptions: We make the following technical assumptions:

A1. The window of true anomaly for data collection from each target and the required attitude pointing remain fixed. This assumption is practical for viewing distant celestial objects over a limited horizon for which the Earth-spacecraft system does not appreciably process along Earth's one-year Solar orbit.

A2. Time required to collect data from each target in terms of corresponding change in the true anomaly of the spacecraft is known but not necessarily constant for all targets.

A3. All critical failures are collapsed into a single failure state. Once in this failure state, the spacecraft can no longer complete its mission.

A4. The probabilities of transitioning to the failure state from any nominal state with execution of any action are known, i.e., the MDP transition probability matrix is known.

A5. Scientific reward associated with fully observing each target is known but not necessarily constant for all targets.

A6. Energy/fuel consumption due to attitude maneuvering or data collection actions is known. Energy consumption is computed based on a pre-specified slewing maneuver profile (see section 4.3.3); data collection energy use is presumed available in tabular form.

Problem Statement: Given a set of *n* targets, our objective is to generate an optimal sequence of attitude slewing maneuvers that enable data collection from observation targets in such a way that a cost function which reflects the expected science reward is maximized while fuel/energy costs are minimized.

Figure 4.1 shows snapshots of a spacecraft orbiting a planet while collecting data from three targets. The triangles represent field of view of the spacecraft while the small solid circles represent distant observation targets. The spacecraft is represented by small rectangles in an elliptical orbit. The pointing of the illustrative observation cone/triangle indicates the attitude of the spacecraft at each true anomaly station along its orbit.

The MDP outputs a policy applicable to any initial state. The policy prescribes the set of actions that yield the highest possible science reward.

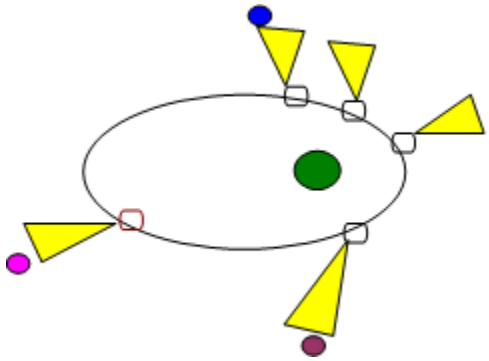

Figure 4.1: Spacecraft at various positions in its orbit collecting data

4.3 MDP Formulation

4.3.1 States

For the attitude planning problem, the states must specify the pointing attitude of the spacecraft, the current true anomaly, and the amount of science data collected as follows:

$$
\begin{aligned}
&S = \{s_1, s_2, s_3, ..., s_N, s_F\} \\
&s_i = \left(b_i^1, b_i^2, ..., b_i^n, z_i, v_i\right) : i \in \{1, 2..., N\} \\
&b_i^j \in \{0,1\}, z_i \in \{0,1,2,...,n\}, v_i \in d\tau, d \in \{1,2,...m\}
\end{aligned} \tag{4.3.1}
$$

In (4.3.1), the binary variable b_i^j indicates whether the data have been collected or not from the j^{th} target in state s_i. The integer z_i represents the pointing of the spacecraft. $z_i = j$ means the spacecraft is pointed towards target j in state s_i or s_i is an initial pointing state (if $z_i = 0$). Variable v_i represents the true anomaly angle of the spacecraft in state s_i. We have discretized true anomaly (0 to 360) into m non-overlapping partitions of size τ ($m = 360/\tau$), selecting τ so that m is an integer. There is a tradeoff between the value of τ (and hence precision of true anomaly) and the size of state space. State s_F indicates that one or more components of the spacecraft have failed in such a manner that the plan can no longer be executed. Multiple failures with partial loss of spacecraft functionality can be treated similarly by introducing additional states reflecting these failure states. The total number of states is $N = 2^n (n+1)m$, where n is the number of science targets.

4.3.2 Actions

Before we define actions, we present a set of true anomaly windows for each of n targets in which the targets are within the range of data collecting equipment:

$$
W = \left\{\left[v_l^1, v_u^1\right], \left[v_l^2, v_u^2\right], ..., \left[v_l^n, v_u^n\right]\right\} \tag{4.3.2}
$$

Here, v_l^k is the lower limit of the window of visibility of target k, while v_u^k is the upper limit of the window of visibility of target k. Besides window of visibility, every target has another attribute that indicates whether the data from the target is to be collected only once in the mission:

$$p = \{p_1, p_2, ..., p_n\} : p_i \in \{0,1\} \tag{4.3.3}$$

Here, n is the number of targets and p_k is a binary variable indicating whether target k requires periodic data acquisition ($p_k = 1$) or not ($p_k = 0$). Actions can then be defined as:

$$\begin{aligned}
&M = \{\mu_0, \mu_1, ..., \mu_n, NOOP\} : \\
&\mu_k(s_i) = \begin{cases} s_j & if : \left(\nu_i + \Delta\nu_{kz_i}\right) < 360 \\ s_q & otherwise \end{cases} : \\
&\nu_j = \nu_i + \Delta\nu_{kz_i}, \nu_q = \left(\nu_i + \Delta\nu_{kz_i}\right) - 360 \\
&i, j, q \in \{1, 2, ..., N\}, z_j = z_q = k, \\
&b_{xj} = b_{xi} \forall x \neq k, \\
&b_{kj} = \begin{cases} 1 & z_i = k \neq 0 \wedge v_l^k \leq \nu_i \leq v_u^k - \Delta\nu_{kz_i} \\ b_{ki} & otherwise \end{cases} \\
&b_{xq} = \begin{cases} b_{xi} & if : p_x = 0 \\ 0 & otherwise \end{cases} : \forall x \neq k \\
&b_{kq} = \begin{cases} 1 & z_i = k \neq 0 \wedge v_l^k \leq \nu_i \leq v_u^k - \Delta\nu_{kz_i} \\ b_{ki} & otherwise \end{cases}
\end{aligned} \tag{4.3.4}$$

where μ_k indicates an action that, when taken from state s_i, results either in state s_j or in state s_q. These states have true anomaly equal to the sum (modulo 360 deg) of the true anomaly of state s_i and the change in true anomaly incurred during the action μ_k. Further, the attitude pointing of state s_j and s_q is k and number of targets visited in state s_j remains the same as the number of targets visited in state s_i unless attitude pointing in state s_i is already k and true anomaly of state s_i is within the window of visibility of target k. For transitioning to s_q, the number of visited targets is reset to zero except for those targets which require only a single visitation. There is one other action denoted as NOOP (no operation) which can be defined as:

$$
\begin{aligned}
&NOOP(s_i) = \begin{cases} s_x & if : (\nu_i + \tau) < 360 \\ s_y & otherwise \end{cases} : \\
&i, x, y \in \{1, 2, ..., N\} \\
&z_x = z_y = z_i \\
&\nu_x = \nu_i + \tau, \nu_y = (\nu_i + \tau) - 360 \\
&b_{kx} = b_{ki} \quad \forall k \in \{1, 2, ..., n\} \\
&b_{ay} = b_{ai} \quad \forall a : p_a = 0 \\
&b_{ty} = 0 \quad \forall t : p_t = 1, a, t \in \{1, 2, ..., n\}
\end{aligned} \tag{4.3.5}
$$

NOOP does not have to be a single action; one can define multiple NOOP actions over different changes in true anomaly as will be described in the examples presented later in this chapter.

To clarify this formulation of actions, we consider an example linear plan described by a sequence of actions that ultimately enable Target 2 to be observed.

Example*:* Action μ_2 from (000, 0, 0) → (000, 2, 35), Action μ_2 from (000, 2, 35)→ (000, 2, 42), Action μ_2 from (000, 2, 42) → (010, 2, 49) → Action NOOP from (010, 2, 49) → (010, 2, 56)

In the above example we assume there are three targets and the change in true anomaly during a slew maneuver from the initial state to Target 2 is 35 deg, while the change in true anomaly during the collection of data from Target 2 is 7 deg. Action 2 can result in a change of true anomaly equal to 35 deg or 7 deg depending upon the state from which it is executed. Also, selecting Action 2 at a true anomaly of 35 deg does not result in immediate collection of data from Target 2 whereas the same action at true anomaly of 42 degrees does result in collection of data due to the fact that the time window for collection of data from Target 2 starts after true anomaly of 35 deg and before or at true anomaly of 42 deg. Finally, NOOP results in true anomaly change only.

4.3.3 Immediate Rewards and Costs

The immediate reward for state s_i is given by:

$$R(s_i) = \left\{ \left(\sum_{k=1}^{n} b_{ki} r_k \right) + \frac{1}{2} r_{z_i} (1 - b_{z_i}) + \alpha \prod_{k=1}^{n} b_{ki} : i \in \{1, 2, ..., N\} \right\} \tag{4.3.6}$$

where b_i (for all i), N, and n are defined in Equation (4.3.1), α is the weighting factor for the reward of collecting all science data, and r_k: $k \in \{1,2\ldots,n\}$ is the reward for the science data expected from target k. Equation (4.3.6) computes immediate rewards for the states based on given rewards for science data at each target. The immediate reward $R(s_i)$ depends upon how much data has been collected on reaching s_i and on the target which the spacecraft is pointed at in s_i. The third factor in Equation (4.3.6) represents the additional reward if the spacecraft succeeds in collecting the data from all the targets ($\alpha > 0$). This term is added to emphasize upon completion of mission that is to collect the data from all the targets. We assume that the immediate reward for failure state, s_F, is zero.

The set of action costs can be represented as:

$$c(\mu_k, s_i) = \begin{cases} \varphi(z_i, k) & z_i \neq k \\ \delta_k & otherwise \end{cases} : z_i, k \in \{0, 1, ..., n\} \tag{4.3.7}$$

where $\varphi(z_i, k)$ is the angle by which the spacecraft (as a rigid body) has to rotate about the axis of rotation. The axis of rotation is typically off-axis (not a pure yaw, pitch, or roll)r to attain the attitude change demanded by action μ_k executed from state s_i. The δ_k in (4.3.7) represents the cost of collecting data from target k. Equation (4.3.7) is an indirect measure of fuel/energy consumed by executing an action. Here, an assumption is made

that energy consumption is proportional to the angle by which the orientation is changed but the cost can be assumed to be constant for each maneuver.

4.3.4 Transition Probabilities

The transition probabilities are defined using the following equation:

$$T(s_i,\mu_k,s_j)=\begin{cases}1-\left(\rho_1\varphi\left(z_i,k\right)\right) & if:\left(z_i\neq z_j\right)\\ \left(1-\rho_2\delta_k\right) & otherwise\end{cases} \qquad (4.3.8)$$
$$\left(\rho_1\varphi\left(z_i,k\right)\right)\leq 1,\rho_2\delta_k\leq 1,i,j\in\{1,2,...,N\}$$

where ρ_1 and ρ_2 are given parameters selected so that the inequality constraints on the second line of Equation (4.3.8) are satisfied in general. In our formulation, every action in every state has two possible outcomes: it could either result in its corresponding desired state (Equation (4.3.4)) or in the failure state. The possibility of ending up in the desired state is given by Equation (4.3.8) while the possibility of failure is 1- $T(s_i, \mu_k, s_j)$. Figure 4.2 illustrates this definition.

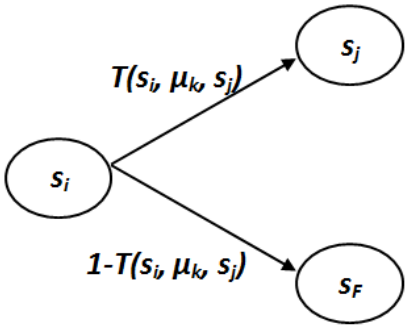

Figure 4.2: State transition map

4.4 Solution Approaches

4.4.1 Infinite Horizon

For computing the optimal policy with respect to an infinite horizon expected discounted reward,

$$E\left[\sum_{t=0}^{\infty}\gamma^t R(s^t,\mu^t)\,|\,Pol,s^0=s\right],$$

the Value Iteration algorithm (Chapter 2) can be used. The advantage of an infinite horizon solution is that the optimal policy can be computed once for the entire mission or for the interval over which science goals and other MDP parameters remain valid. The resulting optimal policy is stationary and assumes fixed reward function and transition probability values.

4.4.2 Finite Horizon

Finite horizon solutions can be produced using a backward induction algorithm. In this case, the function that is optimized is,

$$E\left[\sum_{t=0}^{\kappa}\gamma^t R(s^t,\mu^t)\,|\,Pol,s^0=s\right].$$

The backward induction algorithm is shown in Figure 4.3.

Step 0. Initialization:
Initialize the terminal contribution $V_\kappa(s_\kappa)$
Set $t = \kappa - 1$

Step 1. Calculate

$$V_t(s_t) = R_t(s_t) + \max_{\mu_t}\left\{-C_t(\mu_t) + \gamma \sum_{s' \in S} T_t(s_t, \mu_t, s') V_{t+1}(s')\right\} \quad \text{for all } s_t \in S$$

Step 2. If $t > 0$, decrement t and return to step 1,
else, calculate:

$$P_t^*(s_t) = \max_{\mu_t}\left\{-C_t(\mu_t) + \gamma \sum_{s' \in S} T_t(s_t, \mu_t, s') V_t(s')\right\} \quad \text{for all } t$$

Figure 4.3: Backward Induction algorithm [84]

In this case, the resulting optimal policy is time-varying and the reward function and transition probabilities can also be time varying. Length of horizon κ must be selected carefully to allow completion of mission or to ensure a new policy can be uploaded and prepared for execution before the executing MDP's finite horizon is reached.

4.5 MDP Decomposition-based ADP Approach

To manage computational complexity of the MDP, we present a goal-based approximate dynamic programming algorithm that is inspired by Boutilier's algorithm [14]. The ADP decomposes the full MDP into smaller MDPs, finds the solutions for smaller MDPs, and recombines the resulting optimal value functions of the smaller MDPs to create an estimate of the value function for states of the original MDP. While the approach in [14] is general, our approach is customized for our application with three novel features. First, we do not form decomposed MDPs based on a reward function, rather we decompose the states which allows us to separate out sets of variables required to make decisions related

to data collection from subgroups of science targets as in step 1 of the algorithm below. Second, we do not assume the initial state is known, so our estimated policy is applicable from any initial state (see step 5 below). This is important given uncertainties in launch, deployment, and other disturbances imposed on the spacecraft. Third, we do not perform a search algorithm to merge policies after solving smaller MDPs. Instead, we merge value functions using an addition heuristic and calculate an aggregate policy for the overall MDP using the same equation that is used in the Value Iteration algorithm (see step 4 below). Now, we present our algorithm step-by-step.

Step 0 Define the main MDP with states (S), actions (M), reward function (R), and transition probabilities (T).

Step 1 Create sub-groups of targets based on an appropriate heuristic. For example a reasonable heuristic for creating sub-groups is to group the targets that have data collection windows with-in a particular range of true anomaly intervals.

Step 2 Create smaller MDP's from the main MDP based on the heuristic subgrouping. Assign each of the smaller MDP a sub-group of targets. The following steps can be used for decomposition:

i. The states of each smaller MDP contain only the subset of data collection flags corresponding to the sub-group of targets assigned to that MDP. Attitude pointing and true anomaly variables do not change.

ii. The actions of each smaller MDP should include only the subset of actions corresponding to the attitude pointing and data collection from the sub-group of targets assigned to that MDP.

iii. The reward function and transition probabilities are extracted for each smaller MDP from the main MDP based on the specific states and actions included in that MDP. In the baseline case values remain the same as in the aggregate MDP.

Step 3 Solve each of the smaller MDP's (all with a common finite or infinite horizon) and compute optimal values for the states.

Step 4 Merge the values calculated in step 2 to generate estimates of the optimal values for the main MDP using an addition heuristic as follows:

$$V^{est}(s_i) = V1^*(s1_{j1}) + V2^*(s2_{j2}) + \ldots + Vk^*(sk_{jk})$$

where

$$z_{j1} = z_{j2} = \ldots = z_{jk} = z_i,$$

$$v_{j1} = v_{j2} = \ldots = v_{jk} = v_i.$$

Note that in combining the values it is important to ensure the true anomaly intervals and attitude pointing values for all states of interest match.

Step 5 Calculate the optimal policy estimate using the equation:

$$P^{est}(s) = \max_{\mu} \left\{ -C(\mu) + \gamma \sum_{s' \in S} T(s, \mu, s') V^{est}(s') \right\}$$

for all $s_t \in S$

The following sub-sections discuss the application of this decomposition and recombination strategy using an illustrative example.

4.5.1 MDP Decomposition

Consider an example of six science targets. The states for this example can be written using (4.3.1) as,

$$\begin{aligned} &S = \{s_1, s_2, s_3, ..., s_{32256}, s_F\} \\ &s_i = \left(b_i^1, b_i^2, ..., b_i^6, z_i, \nu_i\right) : i \in \{1, 2..., 32256\} \\ &b_i^j \in \{0,1\}, z_i \in \{0,1,2,...,6\}, \nu_i \in d\tau, d \in \{1,2,...72\} \end{aligned} \tag{4.5.1}$$

Note that the above equation indicates 32,256 normal states due to six binary flags, a pointing variable z with seven possible values, and the true anomaly variable ν with 72 possible values (each value of ν separated by five degrees from the next value). In our decomposition approach, we decompose the state space based on sub-grouping of targets. In this example, we form two sub-groups. The first group consists of targets 1, 2, and 3. The states for this group can be written as,

$$\begin{aligned} &S1 = \{s1_1, s1_2, s1_3, ..., s1_{4032}, s_F\}, \\ &s1_i = \left(b_i^1, b_i^2, b_i^3, z_i, \nu_i\right) : i \in \{1, 2..., 4032\}, \\ &b_i^j \in \{0,1\}, z_i \in \{0,1,2,...,6\}, \nu_i \in d\tau, d \in \{1,2,...72\} \end{aligned}. \tag{4.5.2}$$

Note that, in decomposing the states, we kept the same values for variables z and ν and since we have only three binary flags for indication of collected data, the state space size reduces to 4032. This is because we wish to recombine the value functions later. States for the second group of targets i.e. 4, 5, and 6 can be written in a similar way as in (4.5.2).

$$
\begin{aligned}
&S2 = \{s2_1, s2_2, s2_3, ..., s2_{4032}, s_F\} \\
&s2_i = \left(b_i^4, b_i^5, b_i^6, z_i, \nu_i\right) : i \in \{1, 2..., 4032\} \\
&b_i^j \in \{0,1\}, z_i \in \{0,1,2,...,6\}, \nu_i \in d\tau, d \in \{1,2,...72\}
\end{aligned} \tag{4.5.3}
$$

The set of actions in the first group are

$$M1 = \{\mu_0, \mu_1, \mu_2, \mu_3, NOOP\} \tag{4.5.4}$$

Similarly, the set of actions for the second group are

$$M2 = \{\mu_0, \mu_4, \mu_5, \mu_6, NOOP\} \tag{4.5.5}$$

Note that μ_0 appears in both sets. The actions represent slewing or data collection activities where data is collected only if the spacecraft is already pointed towards the corresponding target and is within the data collection window. On the other hand, an attitude maneuver is executed only if the spacecraft is not already pointed towards the corresponding target as described previously in Section 4.3.2.

Reward and cost functions and transition probabilities for both groups of targets can be extracted from definitions in Sections 4.3.3 and 4.3.4. Subsequently, we refer to the MDP formed by target group {1, 2, 3} as MDP1 and the MDP formed by target group {4, 5, 6} as MDP2.

4.5.2 Recombination of Value Functions

Once the original MDP has been decomposed, the next step is to solve the two smaller MDPs. This will yield optimal value functions $V1^*$ and $V2^*$ for the states of MDP1 and MDP2 respectively. The estimate for the state of the original MDP can be calculated as

$$
\begin{aligned}
&V^{est}(s_i) = V1^*(s1_j) + V2^*(s2_k) \\
&where \\
&z_j = z_k = z_i, v_i = v_j = v_k, \\
&j,k \in \{1,2,...,4032\}, i \in \{1,2,...,32256\}
\end{aligned}
\tag{4.5.6}
$$

Finally, the estimated policy is calculated using

$$
P^{est}(s) = \max_{\mu} \left\{ -C(\mu) + \gamma \sum_{s' \in S} T(s, \mu, s') V^{est}(s') \right\}
\tag{4.5.7}
$$

4.6 Simulation-based Case Studies

4.6.1 General Framework Example

In this section, we consider a spacecraft pointing problem for the case of three targets (*n=3)*. We present three examples to elaborate different tradeoffs in the selection of parameters.

4.6.1.1 Example 1: Effects of Failure Probability Parameters

The states for this example are defined in Equation (4.6.1) where there are 2305 states comprised of one failure state and 2304 normal execution states with all possible combinations of three binary flags for indicating collected science data, the attitude pointing variable z with four possible values and the true anomaly variable v with 72 possible values.

$$
\begin{aligned}
&S = \{s_1, s_2, s_3, ..., s_{2304}, s_F\} \\
&s1_i = \left(b_i^1, b_i^2, b_i^3, z_i, v_i\right) : i \in \{1,2...,2304\} \\
&b_i^j \in \{0,1\}, z_i \in \{0,1,2,3\}, v_i \in d\tau, d \in \{1,2,...72\}
\end{aligned}
\tag{4.6.1}
$$

The set of actions is given by

$$M = \{\mu_0, \mu_1, \mu_2, \mu_3, NOOP\} \tag{4.6.2}$$

The true anomaly windows for the targets were taken as:

$$W = \{[70,120], [280,330], [180,250]\} \tag{4.6.3}$$

The periodic data acquisition indicator set is given by

$$p = \{0,0,0\} \tag{4.6.4}$$

This requires only one data collection activity for each target. Changes in true anomaly incurred during various actions are specified as follows:

$$\begin{aligned} &\Delta\nu_{00} = 5, \Delta\nu_{01} = 50, \Delta\nu_{02} = 70 \\ &\Delta\nu_{03} = 90, \Delta\nu_{12} = 105, \Delta\nu_{13} = 135 \\ &\Delta\nu_{32} = 50, \Delta\nu_{11} = \nu_{22} = \nu_{33} = 10 \\ &\Delta\nu_{ij} = \Delta\nu_{ji} \forall ij \in \{0,1,2,3\} \end{aligned} \tag{4.6.5}$$

In (4.6.5), Δv_{ij} represents change in true anomaly incurred during change in pointing from target i to j which is same as the change in pointing from target j to i. We have seven different NOOP actions each incorporating changes in true anomaly given by

$$\begin{aligned} &\Delta\nu_{NOOP1} = 5, \Delta\nu_{NOOP2} = 50, \Delta\nu_{NOOP3} = 70, \\ &\Delta\nu_{NOOP4} = 90, \Delta\nu_{NOOP5} = 20, \Delta\nu_{NOOP6} = 105, \\ &\Delta\nu_{NOOP7} = 135 \end{aligned} \tag{4.6.6}$$

We can use the above data and Equation (4.3.4) to compute possible outcomes of executing each action in each state. We assumed rewards given by Equation (4.3.6) with

$$r_1 = 30, r_2 = 50, r_3 = 70 \tag{4.6.7}$$

For calculating costs using Equation (4.3.7), we assumed the following angles,

$$\phi_{01} = 50, \phi_{02} = 70, \phi_{03} = 90, \phi_{12} = 105,$$
$$\phi_{23} = 50, \phi_{13} = 135 \tag{4.6.8}$$

where ϕ_{kj} is the orientation angle required for slewing from target k to target j. Also, we set $\delta_k = 1$ for all targets $k \in \{1,2,\ldots,n\}$. We selected $\rho_1 = 0.001$ and $\rho_2 = 10^{-4}$ for calculating transition probabilities in Equation (4.3.8). Also, we assumed $\delta_k = \Delta\nu_{kk}$.

Using the above information and the value iteration algorithm from Chapter 2, we generated an optimal policy after 1395 iterations with discount factor $\gamma = 0.99$. Table 4-1 presents the first ten actions of an optimal trajectory starting from the state (000,0,0). The trajectory in Table 4-1 is unique because each action has only two possible transitions as shown in Figure 4.2.

Table 4-1: Optimal trajectory for example 1.

State	Policy	Outcome	Probability of failure
(000,0,0)	μ_1	(000,1,50)	0.05
(000,1,50)	$NOOP_{50}$	(000,1,100)	0
(000,1,100)	μ_1	(100,1,110)	0.0001
(100,1,110)	μ_2	(100,2,215)	0.105
(100,2,215)	$NOOP_{70}$	(100,2,285)	0
(100,2,285)	μ_2	(110,2,295)	0.0001
(110,2,295)	μ_3	(110,3,345)	0.05
(110,3,345)	$NOOP_{105}$	(110,3,90)	0
(110,3,90)	$NOOP_{90}$	(110,3,180)	0
(110,3,180)	μ_3	(111,3,190)	0.0001

To illustrate the importance of the selection of design parameters, we present results where we change ρ_1 and ρ_2 thereby changing the transition probabilities. Keeping all other information the same as above, we changed ρ_1 to 0.005 and ρ_2 to 0.01 to represent a more risky system. Table 4-2 shows the first ten actions associated with the optimal policy/trajectory from initial state (000,0,0).

Table 4-2: Optimal trajectory with high risk

State	Policy	Outcome	Probability of failure
(000,0,0)	μ_1	(000,1,50)	0.25
(000,1,50)	$NOOP_{20}$	(000,1,70)	0
(000,1,70)	μ_1	(100,1,80)	0.01
(100,1,80)	$NOOP_{90}$	(100,1,170)	0
(100,1,170)	μ_0	(100,0,220)	0.25
(100,0,220)	μ_2	(100,2,290)	0.35
(100,2,290)	μ_2	(110,2,300)	0.01
(110,2,300)	$NOOP_{105}$	(110,2,45)	0
(110,2,45)	$NOOP_{90}$	(110,2,135)	0
(110,2,135)	μ_3	(110,3,185)	0.25
(110,3,185)	μ_3	(111,3,195)	0.01

The distinguishing move for the trajectory in Table 4-2 is taking μ_0 from (100,1,170). This move decomposes a high-risk single action μ_2 into μ_0 and μ_2 with just 15 degrees extra rotation (50+70 as opposed to 105). The probability of failure when executing μ_2 directly from (100,1,90) is 0.525. In Table 4-2, the two actions executed instead have 0.25 and 0.35 probability of failure, respectively. In Figure 4.4 and Table 4-3, we present a comparison between risk taken by the trajectories generated using aggressive and conservative policies from the above cases. In this comparison, the initial states for the trajectories are chosen at random (same initial state for both trajectories in each simulation run). The trajectories run until all science data has been collected. The risk is calculated using the following equation for each simulation run with $\rho_1 = 0.005$ and $\rho_2 = 0.01$.

$$risk = 1 - \prod_{\mu \in traj} T(\varsigma, \mu, \varsigma') \tag{4.6.9}$$

Figure 4.4 shows the cumulative distribution of the difference in risk taken by the two policies for 250 simulations. Table 4-3 shows minimum, maximum, and mean values of

the difference in risks taken by both trajectories over 25, 250 and 2500 simulation runs. Results show that the policy generated for low values of ρ_1 and ρ_2 takes more risk as compared to the policy generated for high values of ρ_1 and ρ_2. This is expected since the policy generated for high risk environment should be more cautious in selecting attitude maneuvers.

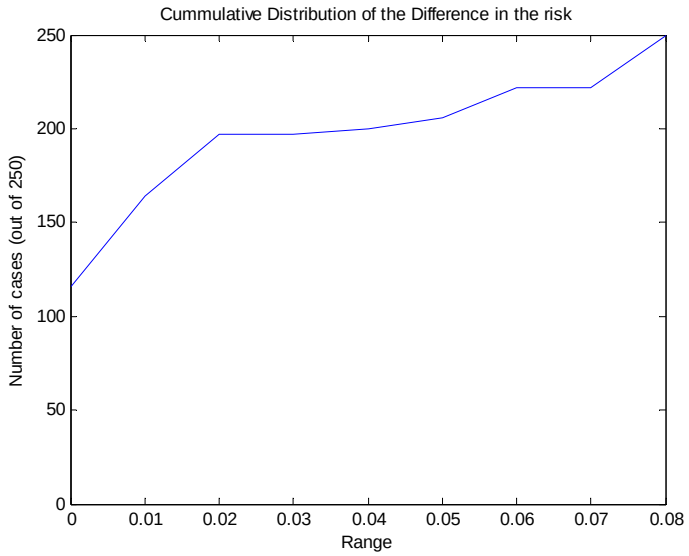

Figure 4.4: Cumulative distribution of the difference in the risk taken by aggressive and conservative trajectories

Table 4-3: Risk comparison (J = $risk_{agg}$ - $risk_{cons}$)

# of Sim.	*min*(*J*)	*max*(*J*)	*mean*(*J*)
25	0	0.0788	0.0199
250	0	0.0788	0.0146
2500	0	0.0788	0.0157

4.6.1.2 Example 2: Effects of Selection of Discount Factor (γ)

This section presents a case where there is a conflict between time windows of the targets so that the spacecraft can collect data from only one out of two conflicting targets. We keep all the data as before except ρ_1 =0.001 and $\rho_2 = 10^{-4}$. The time windows are also changed to:

$$W = \{[50,150],[170,330],[40,130]\}. \tag{4.6.10}$$

Table 4-4 shows first ten actions of an optimal trajectory from initial state (000,0,0). The simulation required 1395 iterations with $\gamma = 0.99$.

Table 4-4: Optimal trajectory for γ = 0.99

State	Policy	Outcome	Probability of failure
(000,0,0)	μ_1	(000,1,50)	0.05
(000,1,50)	μ_1	(100,1,60)	0.0001
(100,1,60)	μ_2	(100,2,165)	0.105
(100,2,165)	$NOOP_{70}$	(100,2,235)	0
(100,2,235)	μ_2	(110,2,245)	0.0001
(110,2,245)	μ_3	(110,3,295)	0.05
(110,3,295)	$NOOP_{105}$	(110,3,40)	0
(110,3,40)	μ_3	(111,3,50)	0.0001
(111,3,50)	$NOOP_5$	(111,3,55)	0
(111,3,55)	$NOOP_5$	(111,3,60)	0

The trajectory with the same parameters and conditions is computed except with γ set to 0.8. This resulted in the optimal policy after only 64 iterations which yielded the following trajectory from initial state (000,0,0).

Table 4-5: Optimal trajectory for γ = 0.8

State	Policy	Outcome	Probability of failure
(000,0,0)	μ_3	(000,3,90)	0.09
(000,3,90)	μ_3	(001,3,100)	0.0001
(001,3,100)	$NOOP_{20}$	(001,3,120)	0
(001,3,120)	μ_2	(001,2,170)	0.05
(001,2,170)	μ_2	(011,2,180)	0.0001
(011,2,180)	$NOOP_{135}$	(011,2,315)	0
(011,2,315)	μ_1	(011,1,60)	0.105
(011,1,60)	μ_1	(111,1,70)	0.0001
(111,3,70)	$NOOP_5$	(111,3,75)	0
(111,3,75)	$NOOP_5$	(111,3,80)	0

In Figure 4.5 and Table 4-6, we present a comparison between the risk incurred by data collection trajectories generated from far sighted (γ = 0.99) and short sighted (γ = 0.80) policies from above. In this comparison, the initial states for the trajectories are chosen at random (same initial state for both trajectories in each simulation run). The trajectories run until all science data has been collected. The risk is calculated using Equation (4.6.9).

Table 4-6: Risk comparison ($J = risk_{SS} - risk_{FS}$)

# of Sim.	*min*(*J*)	*max*(*J*)	*mean*(*J*)
25	0	0.0875	0.0084
250	0	0.1627	0.0070
2500	0	0.1627	0.0077

These results show that short-sighted trajectories are more risky than far-sighted trajectories. Selection of γ therefore maps to a tradeoff between conserving the life of the spacecraft and trying to complete the mission as soon as possible. Since γ is a general parameter of the MDP formulation, this tradeoff should hold for general cases.

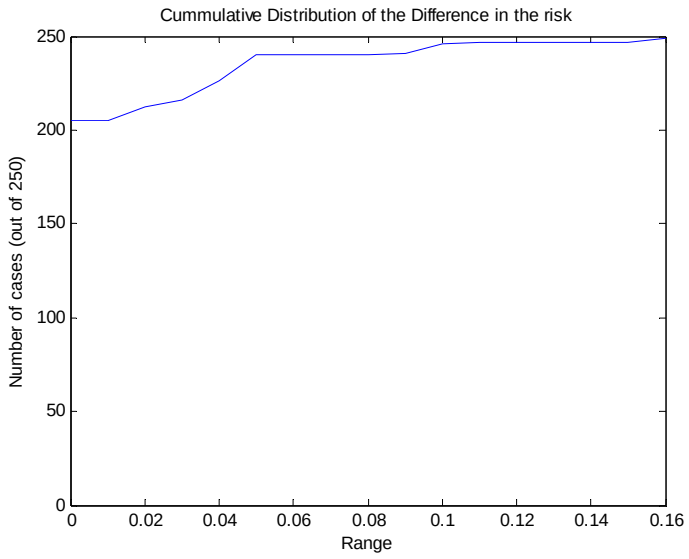

Figure 4.5: Cumulative distribution of the difference is the risk taken by far sighted and short sighted trajectories over 250 simulations.

4.6.1.3 Example 3: Effects of Repeated Targets

We now cast Target 3 as offering reward from repeated data acquisitions i.e. we set $p_3 = 1$ for this example. The time windows for the targets were selected as:

$$W = \{[70,120],[280,330],[180,250]\} \tag{4.6.11}$$

We first computed an optimal policy for the case $\gamma = 0.99$ with 1486 value function iterations. With this policy the trajectory from (000,0,0) had the same first 10 actions as in Table 4-1. Then the trajectory kept on choosing $NOOP_5$ until it reached (111,3,355). From there, the trajectory had the actions listed in Table 4-7 to enter the cycle.

Table 4-7: Optimal trajectory for $\gamma = 0.99$

State	Policy	Outcome	Probability of failure
(111,3,355)	$NOOP_{50}$	(110,3,45)	0
(110,3,45)	$NOOP_{135}$	(110,3,180)	0
(110,3,180)	μ_3	(111,3,190)	0.0001

Since this example simulates a low-risk scenario, the policy tends to minimize the number of actions required to collect data. Note that the data from all the targets to be visited one-time were collected in the first revolution and then the spacecraft remained pointed towards the target requiring repeated data acquisition. With $\gamma = 0.8$, the effective planning horizon is reduced. The resulting policy had the following trajectory leading to a cycle.

Figure 4.6 and Table 4-9 show a comparison between risks taken by the trajectories generated using periodic and aperiodic trajectories with $\gamma = 0.99$. In this comparison, the initial states for the trajectories are chosen at random (same initial state for both

trajectories in each simulation run). The trajectories run until all science data has been collected. The risk is calculated using Equation (4.6.9).

Table 4-8: Optimal trajectory for $\gamma = 0.8$

State	Policy	Outcome	Probability of failure
(000,0,0)	μ_1	(000,1,50)	0.05
(000,1,50)	$NOOP_{20}$	(000,1,70)	0
(000,1,70)	μ_1	(100,1,80)	0.0001
(100,1,80)	μ_3	(100,3,215)	0.135
(100,3,215)	μ_3	(101,3,225)	0.0001
(101,3,225)	$NOOP_5$	(101,3,230)	0
(101,3,230)	μ_2	(101,2,280)	0.05
(101,2,280)	μ_2	(111,2,290)	0.0001
$NOOP_5$ until (111,2,355)			
(111,2,355)	μ_3	(110,3,45)	0.05
(110,3,45)	$NOOP_{135}$	(110,3,180)	0
(111,3,180)	μ_3	(111,3,190)	0.0001
$NOOP_5$ until (111,3,355)			

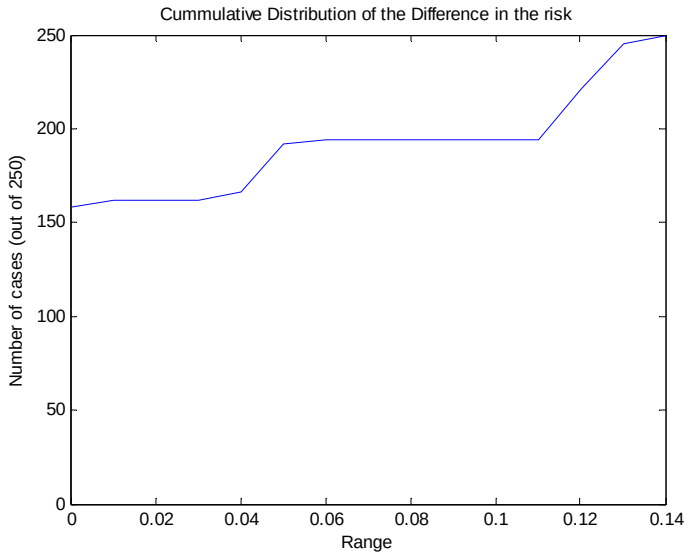

Figure 4.6: Cumulative distribution of the difference is the risk taken by periodic and aperiodic trajectories over 250 simulations.

Table 4-9: Risk comparison ($J = risk_{Periodic} - risk_{Aperiodic}$)

# of Sim.	*min*(*J*)	*max*(*J*)	*mean*(*J*)
25	0	0.1215	0.0412
250	0	0.1359	0.0348
2500	0	0.1359	0.0339

It is interesting to note that in this case periodic trajectories take more risk as compared to aperiodic trajectories. This result should be true for general cases since periodic targets may require additional attitude maneuvers since multiple time data collection is required for such targets.

4.6.2 Approximate Dynamic Programming Example

In this section we present simulation results for the example introduced in Section 4.5. We used same NOOP actions as in (4.6.6). The true anomaly windows for the six targets are given by:

$$W = \begin{Bmatrix} [50,100], [100,150], [150,200], \\ [200,250], [250,300], [300,350] \end{Bmatrix} \tag{4.6.12}$$

All targets are aperiodic (i.e. require only one-time data collection). Changes in true anomaly incorporated during various actions were specified as follows:

$$\begin{aligned} &\Delta\nu_{00} = 5, \Delta\nu_{01} = 50, \Delta\nu_{02} = 70, \Delta\nu_{03} = 90, \\ &\Delta\nu_{04} = 80, \Delta\nu_{05} = 40, \Delta\nu_{06} = 60, \Delta\nu_{12} = 105, \\ &\Delta\nu_{13} = 135, \Delta\nu_{14} = 150, \Delta\nu_{15} = 120, \Delta\nu_{16} = 80, \\ &\Delta\nu_{23} = 50, \Delta\nu_{24} = 70, \Delta\nu_{25} = 100, \Delta\nu_{26} = 130, \\ &\Delta\nu_{34} = 40, \Delta\nu_{35} = 70, \Delta\nu_{36} = 90, \Delta\nu_{45} = 50, \\ &\Delta\nu_{46} = 80, \Delta\nu_{56} = 30, \\ &\Delta\nu_{11} = \Delta\nu_{22} = \Delta\nu_{33} = \Delta\nu_{44} = \Delta\nu_{55} = \Delta\nu_{66} = 10 \\ &\Delta\nu_{ij} = \Delta\nu_{ji} \forall ij \in \{0,1,2,3,4,5,6\} \end{aligned} \tag{4.6.13}$$

In (4.6.13), Δv_{ij} represents change in true anomaly incurred during change in pointing from target i to j which is same as from target j to i. Rewards are given by Equation (4.3.6) with the following parameter values:

$$r_1 = 30, r_2 = 50, r_3 = 70, r_4 = 40, r_5 = 60, r_6 = 80 \tag{4.6.14}$$

For calculating costs using Equation (4.3.7), we assume the following angles,

$$\begin{aligned}
&\phi_{01} = 50, \phi_{02} = 70, \phi_{03} = 90, \phi_{04} = 80, \\
&\phi_{05} = 40, \phi_{06} = 60, \phi_{12} = 105, \phi_{13} = 135, \\
&\phi_{14} = 150, \phi_{15} = 120, \phi_{16} = 80, \phi_{23} = 50, \\
&\phi_{24} = 70, \phi_{25} = 100, \phi_{26} = 130, \phi_{34} = 40, \\
&\phi_{35} = 70, \phi_{36} = 90, \phi_{45} = 50, \phi_{46} = 80, \\
&\phi_{56} = 30
\end{aligned} \tag{4.6.15}$$

Here, ϕ_{kj} is the slew maneuver (angle) required for changing pointing from target k to target j. Also, we set $\delta_k = 1$ for all targets $k \in \{1,2,\ldots,n\}$. We select $\rho_1 = 0.001$ and $\rho_2 = 10^{-4}$ for calculating transition probabilities based on (4.3.8), and we assume $\delta_k = \Delta v_{kk}$.

Optimal values were generated for the states of both MDPs using an infinite horizon value iteration algorithm with $\gamma = 0.99$. The values were merged using the ADP addition heuristic from Equation (4.5.6) i.e.,

$$V^{est}(s) = V_1^*(s_1) + V_2^*(s_2), \tag{4.6.16}$$

where s_1 and s_2 are states in the decomposed MDPs that combine to form state s in the original MDP. For example, the state (010, 5, 255) (in group 1) and state (100, 5, 255) (in group 2) combine to form the state (010100, 5, 255) in the main MDP. Finally, we compute the estimated policy P^{est} using (4.5.7) with estimated values. We also solve the main MDP and compute policy P^* from optimal values.

To compare the two policies, we carried out simulations in which trajectories were computed using random initial state for both policies. The stopping criterion for a trajectory was an execution of 200 actions after acquisition of data from all six targets by both policies and five additional actions (to incorporate steady state behavior and account for the fact that optimization is over infinite trajectories). Results are shown in Figure 4.7 and Table 4-10. The normalized expected values shown in Figure 4.7 and their difference as shown in Table 4-10 for all simulations were calculated according to the following equations:

$$
\begin{aligned}
&ev1(i) = \sum_{t=0}^{k} \gamma^t E\left[R(s_t) - C(\mu_t)\right] : ith - simulation \\
&EV1 = [ev1(1) \quad ev1(2) \quad \ldots \quad ev1(x)] : x - simulations \\
&ev2(i) = \sum_{t=0}^{k} \gamma^t E\left[R(w_t) - C(\vartheta_t)\right] : ith - simulation \\
&EV2 = [ev2(1) \quad ev2(2) \quad \ldots \quad ev2(x)] : x - simulations \\
&\Delta EV = \frac{EV1 - EV2}{\max(\max(EV1), \max(EV2))} \\
&w_0 = s_0
\end{aligned}
\tag{4.6.17}
$$

Table 4-10: Comparison of P^* and P^{est}

No. of Simulations	ΔEV_{max}	ΔEV_{min}	ΔEV_{mean}
25	0.0459	0	0.0150
250	0.0750	0	0.0182
2500	0.1203	0	0.0189

The results show that the ADP-based policy attains on average 98% of the optimal reward attained by the optimal policy. The computational complexity of the original MDP is of the order 10^{10}, whereas the computational complexity of the ADP-based MDP is of the order 10^8, two orders of magnitude less. Figure 4.7 shows that for most states the percentage difference in the expected values obtained by the optimal and the estimated

policies is quite low. In fact the maximum percentage difference for all possible states (32,256) is 13.6. This verifies the results presented in Table 4-10.

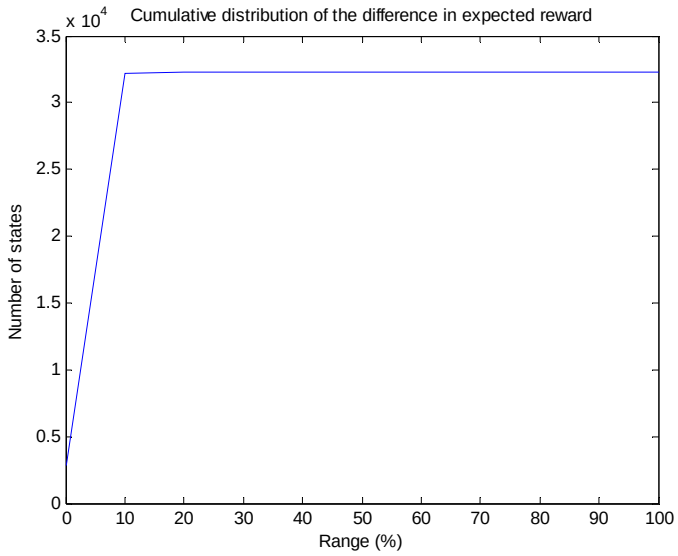

Figure 4.7: Cumulative distribution of the percentage difference in expected values obtained by trajectories of P^* and P^{est}

4.6.3 Further Analysis of the Approximate Dynamic Programming Example

In this section, we present the performance results of the approximate dynamic programming example from Section 4.6.2. To be precise, we change the values for different parameters in the planning MDP and record the performance of the ADP-based MDP. The expected reward for simulated trajectories with each possible initial state is then compared for ADP-based MDP against the original MDP. Table 4-11 shows the results for variations in the epoch (horizon length), discount factor (γ), weighting factor for goal state reward (α), risk factor (ρ_1), and level of overlap for the data collection windows of the targets. Note that the level of overlap ranges between 1 and 10 where level 10 is the least overlap. Data collection windows for level 10 are the same as in Section 4.6.2 and for every level below level 10 the windows for targets 4, 5, and 6 are shifted back in true anomaly by 15 degrees. This means that for the level 1 overlap, the

data collection windows for targets 4, 5, and 6 are [65,115], [115,165], and [165,215] respectively. Recall that the data collection windows for targets 1, 2, and 3 are [50,100], [100,150], and [150,200] respectively.

Table 4-11: Performance results for the ADP-based MDP

Variable Factor	**Constant Factors**	**Best Average performance**	**Worst Average Performance**	**Worst case Performance**
Epoch: 3-50	$\gamma = 0.99$ $\alpha = 1000$ $\rho_1 = 0.001$ Level of overlap = 1	99.9% for Epoch = 3	93.5% for Epoch = 8	34% for Epoch = 10
γ = 0.29:0.1:0.99	$\alpha = 1000$ $\rho_1 = 0.001$ Epoch = 50 Level of overlap = 1	99.9% for γ = 0.29	96.7% for γ = 0.79	54% for γ = 0.79
α = 100:500:5000	$\gamma = 0.99$ Epoch = 50 $\rho_1 = 0.001$ Level of overlap = 1	99.3% for α = 600	98.9% for α = 100	88% for α = 100
ρ_1 = 0.001:0.01:0.1	$\gamma = 0.99$ Epoch = 50 $\alpha = 600$ Level of overlap = 1	99.3% for ρ_1 = 0.001	62.1% for ρ_1 = 0.021	9.6% for ρ_1 = 0.071
ρ_1 = 0.001:0.01:0.1	$\gamma = 0.99$ Epoch = 11 $\alpha = 1000$ Level of overlap = 1	95.2% for ρ_1 = 0.001	87.6% for ρ_1 = 0.021	31.5% for ρ_1 = 0.051
Level of Overlap = 1-10	$\gamma = 0.99$ Epoch = 11 $\alpha = 1000$ $\rho_1 = 0.07$	92.4% for Level of overlap = 1	90.5% for Level of overlap = 10	28.7% for Level of overlap = 10
Level of Overlap = 1-10	$\gamma = 0.99$ Epoch = 50 $\alpha = 1000$ $\rho_1 = 0.001$	99.3% for Level of overlap = 1	94.4% for Level of overlap = 9	60.3% for Level of overlap = 9

The worst-case results in Table 4-11 are for Epoch = 10 where the variable factor is Epoch and for $\rho_1 = 0.071$ where the variable factor is ρ_1. Upon further analysis of these cases, we found that Epoch = 10 has bad performance because the policy generated using the original MDP is able to complete the mission in 10 steps and hence collects additional reward corresponding to mission completion (see Equation (4.3.6)) whereas the policy generated using the ADP-based MDP is unable to complete the mission. For lower or higher values of the Epoch, either both MDP policies are able to complete the mission or both are unable to complete the mission and hence the performance of the policy generated using the ADP-based MDP is reasonable for those values of the Epoch. In the case of $\rho_1 = 0.071$, a similar situation arises. In this case, the policy generated using the ADP-based MDP is unable to complete the mission because the value of risk factor is too high to enable a sufficient set of attitude maneuvers to complete the mission whereas the same value of ρ_1 does not prevent the original MDP from completing the mission. This suggests that the policy generated using our proposed ADP approach is conservative (at least in case of our case study) as compared to the original optimal policy. Also for higher or lower values of ρ_1, where either both MDP policies complete the mission or both MDP policies do not complete the mission due to high risk, the performance of the ADP-based MDP policy is reasonable.

4.7 Alternate Formulations and Complexity

This chapter presents one of many possible MDP formulations for spacecraft mission planning. Every mission has its own specific requirements and specifications that likely result in complexities in the state-space beyond decomposing the mission into a set of independent objectives. For example, some missions may require management of energy

resources while others might require the decision about when to communicate with the ground station. There also might be factors such as pointing towards or away from the sun under certain circumstances, variable costs of the mission-related actions, occurrences of exogenous events that might change the mission objectives and/or their relative importance e.g. science rewards etc. Regardless of additional model attributes, the MDP remains a general-purpose tool for spacecraft activity planning. The ADP, potentially with analogous decomposition over true anomaly observation windows, is still a viable approach to reducing complexity to manageable levels.

Once the problem is formulated as an MDP, it can be solved for finite-horizon optimal policies using backward induction [84]. Backward induction allows the reward and cost functions and the transition probabilities to be time-varying. When using backward induction, it is important to select an optimal horizon length that allows for the completion of the mission even in the presence of certain faults. Another important issue is the selection of the state space. A careless selection may lead to unnecessarily large state space size making it difficult to compute the solution. Often there are weak links between the decisions and the information required to make those decisions that create a chance to decompose a large MDP into smaller MDPs without losing much optimality. Such situations of reducing the computational complexity should be considered. Defining the actions to be context-dependent sometimes also reduces state-space size. For example, in our formulation, we used only one action for both data collection and attitude maneuvering. Options like this should be explored based on the structure of the problem at hand.

4.8 Conclusions

We have modeled the problem of autonomous planning of spacecraft pointing sequences based on the theory of Markov Decision Processes. Our objectives were to maximize the science reward, account for action cost (such as energy consumption) and treat emerging failure states. We have also presented a method to implement our framework in a finite receding horizon scenario and an approximate dynamic programming formulation to reduce complexity for large-scale problems.

The presented framework accounts for one-time and repeat-visit targets and incorporates observation time window constraints. Our case studies demonstrate that changing the risk probability strongly influences the control policy from maximizing more immediate rewards to reducing risks. We also demonstrate that the policy generated by using our framework can deal with conflicts in target observation windows while maximizing science rewards. In the case of sufficiently long planning horizon, the optimal policy resulting from our models and cost assumptions is to collect data from one-time targets first before collecting data from repeat-visit targets. However, if the effective planning horizon is reduced by decreasing the discount factor then the optimal policy equally emphasizes collecting data from repeated targets as from single visit targets.

In this chapter we also presented an example of approximate dynamic programming which indicated that the policy calculated using approximate value functions using our proposed ADP algorithm can perform almost as well as the policy calculated from optimal integrated value functions.

In future work we will develop a formulation that incorporates re-planning and contingency planning based on observed failures that cause the current planner to halt its executing plan (i.e., safe the spacecraft). Also we anticipate that in future spacecraft missions, the spacecraft can be made more robust by learning over time the values of rewards, risks, and even transition probabilities.

Chapter 5

Conflict Resolution Algorithms and Collaborative Fault Detection

This chapter first describes two algorithms for conflict resolution between two fault detection schemes then uses these algorithms for collaborative fault detection. In our first conflict resolution method, we assume initially that there is no conflict and optimize detection thresholds of both fault detection schemes with respect to a partial cost function that penalizes false alarms and missed detections. Then we continuously update thresholds based on a comprehensive cost function that penalizes conflicts in addition to false alarms and missed detections. Our updates are bounded and managed in such a way that the cost function always assumes the lowest possible cost as a function of thresholds. We make use of residual signals to minimize computational complexity.

In our second conflict resolution method, we present a different solution to the conflict resolution problem using a Markov Decision Process framework that generates an optimal policy for adjusting the fault detection thresholds. This method is computationally more complex but it is more general, does not require knowledge of residuals, and does not require initial optimization of the thresholds. We introduce an error signal that indicates failure in resolving the conflict using threshold updating in which case, a supervisor (human or computer) can be alerted and prompted to take a corrective action. We illustrate our methods on a spacecraft attitude control thruster-valve

system simulation with high noise. Our results show good performance and substantial reduction in conflicts under highly uncertain conditions.

In the second part of this chapter (sections 5.8 through 5.14), we present a framework based on MDP for facilitating the implementation of collaborative fault detection through conflict resolution. The conflict arises when two fault detectors make opposite decisions about the presence of a fault. Transition probabilities for various modes of the components within the system are represented by a Bayes Network. The transition probabilities for some fault flags given applied changes in thresholds are assumed to be pre-calculated using Monte Carlo simulations or other similar methods. Since MDP suffers from the curse of dimensionality, we also present an approximate dynamic programming (ADP) approach for our framework based on decomposition and recombination of states. A comprehensive example is included to demonstrate the implementation of the proposed framework and corresponding ADP approach.

5.1 Conflict Resolution Algorithms

Autonomous aerospace systems require increasingly sophisticated fault protection systems that maximize their ability to maintain a safe operational state in the presence of onboard system failures or environmental anomalies that pose risk or degrade performance. Several strategies have been proposed [99][75][111] to detect, diagnose, and reconfigure in the presence of faults. The Markov Decision Process (MDP) and variants have been considered to manage discrete system models [75][104], while signal filtering, system identification, and adaptive control algorithms have been developed to manage physics-based (continuous) system models [111]. While many of the decisions

made by discrete versus physics-based deliberation engines are distinct, many decisions or conclusions can also impact or overlap with the others [70].

Considering the cost of space missions, their associated communication constraints, and the amount of risk involved due to hostile and uncertain deep space environment, it is desirable for space missions to have multiple fault detection schemes. In this situation, two or more detection schemes may occasionally render inconsistent decisions about the occurrence of a fault. Therefore, a conflict resolution algorithm is desirable.

This chapter presents a formal language and protocol by which symbolic and physics-based fault management systems can share information to negotiate consistent decisions with respect to fault detection. Specifically, we present two methods of conflict resolution that minimize or eliminate discrepancies between the fault information obtained from two separate fault detection algorithms. Our methods apply to any pair of detection algorithms that satisfy corresponding assumptions. Our first method is based on initial threshold optimization with respect to a partial objective function and subsequent threshold updating that is optimal with respect to a specified cost function. While optimizing the thresholds, we make use of residual signals to minimize computational complexity. If the resulting minimum value of the objective function allows a persistent unresolved conflict, an error flag is generated that can be used to alert a human supervisor. The updating equations for thresholds attempt to keep the thresholds as close to the optimal values as possible without causing a conflict, with changes optimized within bounds imposed to achieve minimum acceptable performance criteria.

Our second conflict resolution method is based on a Markov Decision Process (MDP). This approach makes use of the reward function and discount factor to optimize changes

in fault detection thresholds. This method does not require knowledge of residual signals but it is also more computationally-intensive.

In this chapter we apply our conflict resolution strategy to a spacecraft example in which we model continuous time dynamics of the spacecraft and associated faults, as well as a limited number of discrete parameters (e.g., instrument on/off, valve open/shut). Our first fault detection scheme is based on an Interacting Multiple Model (IMM) framework that uses multiple models for the spacecraft to represent dynamics associated with certain specific fault conditions. With this strategy a bank of observers use the sensor data to compute the residuals for each fault models. The model with the lowest residual is assumed to be the true model and the fault condition that it relates to is considered to be the true condition of the spacecraft. Our second fault detection scheme is based on state transition system [104] with Markov assumption. Fault detection is based on the likelihood of reaching failure states given the transition probability table.

In the next sections, we define the problem and present the two fault detection schemes in the context of a limited spacecraft fault detection model. In Section 5.5, we present our threshold optimization and updating methods. Section 5.6 shows simulation results.

5.2 Basic Threshold Adjustment Approach to Conflict Resolution

To motivate subsequent developments, we consider a system that uses two schemes to detect a particular fault. We assume that the performance of each scheme is represented by the probabilities of missed detection *P(MD)* and false alarm *P(FA)*, which are functions of adjustable parameters or thresholds associated with each scheme. Let $J(v^1, v^2)$ be a risk-based cost function which determines the combined performance of

two fault detection schemes (denoted 1 and 2) as a function of two scalar fault parameters v^1 and v^2, one parameter for each scheme,

$$J(v^1, v^2) = a_-^1 P\left(FA^1 \mid v^1\right) + a_+^1 P\left(MD^1 \mid v^1\right) + a_-^2 P\left(FA^2 \mid v^2\right) + a_+^2 P\left(MD^2 \mid v^2\right) \tag{5.2.1}$$

Here $a_-^1, a_+^1, a_-^2, a_+^2$ are positive weights that can be adjusted to emphasize missed detection and false alarms of either scheme. Note that $J(v^1, v^2)$ is a separable function of its arguments, i.e., it can be represented as $J(v^1, v^2) = J_1(v^1) + J_2(v^2)$.

Suppose now the two schemes make calls regarding the presence or absence of a particular fault given the vector of current inputs and operating conditions, U. The fault flags of the two schemes are denoted by $b^1(v^1, U^1), b^2(v^2, U^2)$. The fault flag functions take binary values, either 0 or 1, depending on the inferred absence or presence of a fault.

The existence of a conflict corresponds to a situation in which $b^1(v^1, U^1) \neq b^2(v^2, U^2)$, i.e., one of the schemes indicates a fault and the other does not. To resolve an apparent conflict, a fault or no fault decision needs to be made. Such a call can be made by adjusting v^1 and v^2 so that $b^1(v^1, U^1) = b^2(v^2, U^2)$ and J is minimized.

$$\begin{aligned} & J(v^1, v^2) \rightarrow \min_{v^1, v^2} \\ & s.t. \\ & b^1(v^1, U^1) - b^2(v^2, U^2) = 0 \end{aligned} \tag{5.2.2}$$

Due to the discontinuous nature of the fault flags, $b^1(v^1, U^1)$ and $b^2(v^2, U^2)$, which take binary values, the above optimization problem as stated can only be solved by a systematic grid search.

The case in which an easier solution strategy can be defined is when the parameter v^i is an additive threshold, i.e., the fault flags satisfy

$$b^i(V^i, U) = 0 \text{ iff } \mathrm{h}^{\mathrm{i}}(U^i) - v^i \leq 0 \quad (\mathrm{i} = 1,2) \tag{5.2.3}$$

Here, $h^i(U^i)$ are outputs (smooth functions) that we compare against the thresholds, and $h^i(U^i) - v^i$ are residuals. In case of multiple thresholds, where v^i and $h^i(U^i)$ are vectors, the inequalities in (5.2.3) are understood in a component-wise sense. In this case, given that the objective function $J(v^1, v^2)$ is separable, the original optimization problem reduces to a finite number of smooth optimization problems that can be solved numerically. Modifications of these ideas will be used in the subsequent sections to define and illustrate two conflict resolution schemes.

5.3 Problem Formulation

We now discuss specific assumptions about the two fault detection schemes for which conflicts are to be resolved.

A1. Both detection schemes use numerical thresholds that determine the values of fault flags based on the information available and the inputs to the detection schemes.

A2. The residual signal which is the difference between an output and a threshold, based on which the fault flag is set, is known.

A3. The probabilities of false alarm (FA) and missed detection (MD) for each overlapping fault in both detection schemes are known (not necessarily analytically) and are monotonic functions of thresholds.

A4. The communication of the information between the conflict resolution and fault detection schemes, and computations, are instantaneous.

These assumptions are justified for many practical detection schemes. *Problem Statement:* Under the assumptions stated above, devise an algorithm for conflict resolution between two fault detection schemes to reach consensus with minimum probabilities of false alarms and missed detections.

Below, we first present fault detection algorithms for a spacecraft. We then describe conflict resolution methods and apply them to the spacecraft case study.

5.4 Example Fault Detection Schemes: A Spacecraft Case Study

We now present a spacecraft case study to illustrate the use of two fault detection schemes and associated need for a conflict resolution strategy. One of the fault detection schemes makes use of the physics-based dynamics model and the other makes use of a qualitative, logic-based model.

Consider a 1 DOF satellite attitude control system with two thrusters as shown in Figure 5.1. The dashed line shows the axis of rotation. Two thrusters are mounted in such a way that they produce equal and opposite forces resulting in torque about the axis of rotation.

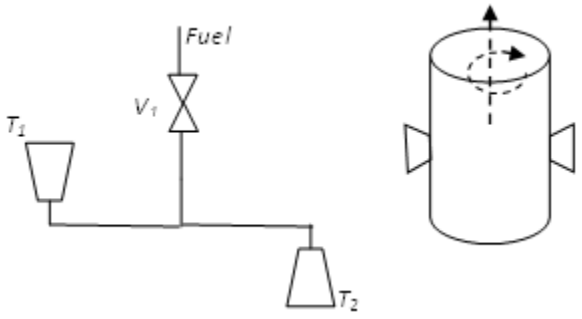

Figure 5.1: 1 DOF satellite schematic

The equations of motion for this system in normal (no-fault) mode can be written as

$$\begin{aligned} x(k+1) &= Dx(k) + Eu_c(k) + G\varepsilon(k), \\ y(k) &= Lx(k) + N\eta(k). \end{aligned} \tag{5.4.1}$$

Here, state vector x represents orientation and angular velocity of the spacecraft, $u_c \in \{0, 1\}$ represents an impulsive thrust value of off (0) or on (1); y represents sensor readings, ε represents system disturbances, and η represents sensor noise. Matrices D, E, G, L, and N are assumed to be of appropriate dimensions. We assume disturbance and noise are normally distributed with zero mean values and known variances. Our physics-based fault detection technique relies on the Interacting Multiple Model (IMM) approach [112]. In this approach, we make use of the fact that, under certain faults (or combination of faults), the system has a specific and known dynamic model. Transitions between the set of possible dynamics models can be treated as discrete jumps. In this chapter we consider only one fault case, i.e. thrust failure. This leads to a discrete state *m(k)* taking values in state set $S = 0,1$. At each decision step k, transition probability $\pi_{ij}(k)$ of the model can be defined by

$$\begin{aligned} &\pi_{ij}(k) = P\{m(k) = j \mid m(k-1) = i\} \quad \forall i, j \in S \\ &\sum_{j \in S} \pi_{ij}(k) = 1 \quad i = 0,1 \end{aligned} \tag{5.4.2}$$

Now, consider a system model representing the fault states plus a nominal operation state (0).

$$\begin{aligned} &x(k+1) = D_j(k)x(k) + E_j(k)u_c(k) + G_j(k)\xi_j(k) \\ &y(k) = L_j(k)x(k) + N_j(k)\eta_j(k) \\ &j = 0,1 \end{aligned} \tag{5.4.3}$$

With the following values for disturbance and noise covariance:

$$\begin{aligned}&\xi_j(k)=\mathbf{N}[\bar{\xi}_j(k),Q_j];\eta_j(k)=\mathbf{N}[\bar{\eta}_j(k),R_j];x_j(0)=\mathbf{N}[\bar{x}_0,\bar{P}_0]\\&\bar{\xi}_j=\bar{\eta}_j=0,Q_j=0.01I_2,R_j=0.05I_2,\bar{P}_0=10I_2\quad\forall j\end{aligned}. \tag{5.4.4}$$

Each cycle of IMM-based fault detection consists of four steps: mixing of estimates, model-conditional filtering, mode probability update and fault detection and diagnostics (FDD) logic, and combination of estimates [112]. Details of these steps for *n*-fault case are presented in [112]. Fault flag *b* is set based on whether or not the likelihood of a mode has crossed corresponding threshold v^1.

$$b(k)=\begin{cases}1 & if: P\{m(k)=1\}\geq v^1\\0 & if: P\{m(k)=1\}<v^1\end{cases} \tag{5.4.5}$$

For our spacecraft, we define candidate probability functions for missed detection (MD) and false alarm (FA) for IMM. As mentioned earlier, we assume that these functions are monotonic with the value of v^1 (see Figure 5.2).

$$\begin{aligned}P(FA\,|\,v)&=\frac{1}{1+e^{10(v^1-0.4)}}\\P(MD\,|\,v)&=\frac{1}{1+e^{-10(v^1-0.6)}}\end{aligned} \tag{5.4.6}$$

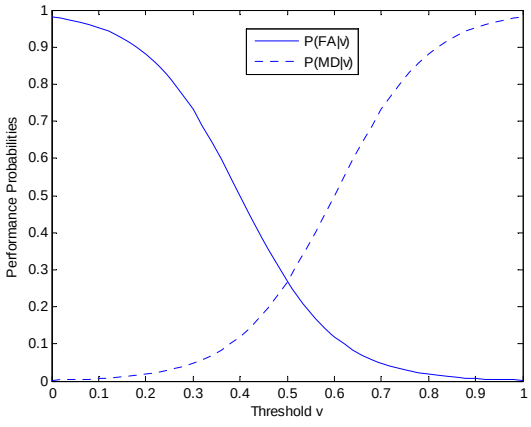

Figure 5.2: Example MD and FA probabilities versus threshold for IMM

Our second fault detection scheme for the same 1 DOF spacecraft is based on a qualitative model of the system that models components, their composition, and possible discrete value sets. As shown above in Figure 5.1, the main spacecraft components of interest are valves and thrusters. For fault detection purposes, we define the following logical clauses to describe the healthy system:

$$\begin{aligned} &(i) \quad V_1 \vee \neg T_1 \vee \neg T_2, \\ &(ii) \quad T_1 \wedge T_2 \vee \neg V_1, \\ &(iii) \quad \neg T_1 \vee T_2, \\ &(iv) \quad T_1 \vee \neg T_2. \end{aligned} \tag{5.4.7}$$

Here, V_1 represents that the corresponding valve is *open* and its complement $\neg V_1$ represents a *closed* valve. Similarly T_i represents that thruster *i* is *on* and its complement $\neg T_i$ represents an *off* thruster. In this model, we make use of the facts that the valve must be *open* for the thrusters to be *on* and the thrusters operate as a pair. This model can detect faults based on sensor readings. To identify faults, we may use the scheme presented by Williams et al [104]. We define the system by triplet $S = (\Pi, \Sigma, \Omega)$ for valve V_1 where Π denotes the set of possible state features, Σ is the set of possible feature value sets, and Ω is a finite set of transitions between states. In our example, we have

$$\begin{aligned} &\Pi = \{status, cmdin, senout\}, \\ &\Sigma = \{\{normal, failed\}, \{open, close, none\}, \{open, close, none\}\}, \\ &\Omega = \{\tau_1, \tau_2, ..., \tau_{(2\times3\times3)\times(2\times3\times3)}\}. \end{aligned} \tag{5.4.8}$$

Each transition is characterized by transforming the state variables from one set of values to the same set or any other set reachable through a transition in Ω. Thruster states are a function of valve states; therefore we do not model thrusters with separate transitions. For each given state configuration, there is a set of possible transitions with associated

probabilities, where the sum of all probabilities is equal to 1. This leads to the transition probability table of size 18×18 which we assume to be known.

If we represent O_t as the set of observations at time t and μ_t as the set of possible commands or actions, we can obtain the set of feasible states at time $t+1$ as

$$S_{t+1} = \left(\bigcup_j \tau_j \left(S_t \bigcap S_{\mu t} \right) \right) \bigcap \Sigma \bigcap S_{Ot+1}. \tag{5.4.9}$$

Once, S_{t+1} is computed, we can determine the most likely trajectories using Bayes rule,

$$P(\tau \mid O_t) = \frac{P(O_t \mid \tau) P(\tau)}{P(O_t)}. \tag{5.4.10}$$

In Equation (5.4.10), if $\tau(S_{t-1})$ and O_t are disjoint sets then clearly $P(O_t| \tau) = 0$. Similarly, if $\tau (S_{t-1})$ is a proper subset of O_t then O_t is entailed and $P(O_t| \tau) = 1$, and hence the posterior probability of τ is proportional to the prior. If neither of the above two situations arises then $P(O_t| \tau) < 1$. Estimating this probability is intricate and requires more research [104]. Finally, the best estimate of current state is found using conflict-directed best first search. Once the conclusion is made about the current state of the valve, fault flag b for the fault is set (1) if a valve has *status = failed* and is cleared (0) otherwise. Failure status is computed from observation O and hence the thresholds since the observed values depend upon comparisons of sensor values against the thresholds.

$$senout = \begin{cases} open & if : SensedVoltage > v^2, \\ close & if : SensedVoltage < v^2, \\ none & if : SensorFailed. \end{cases} \tag{5.4.11}$$

This scheme can detect thruster failures based on valve failures. For example, let us consider the case where valve V_1 is used to supply fuel to thrusters. If *cmdin = open* but *senout = close* after an appropriate delay from initiation of the open instruction then the

valve is failed in *close* mode; note that in this simplified model we assume a failed sensor will reliably give a *SensorFailed* status. The probabilities in Figure 5.3 depend upon the threshold, control command, and the previous state of the valve itself. For conditions when probabilities change with threshold i.e. trying to open a closed valve or close an open valve, we can model the probabilities as functions of threshold in a similar way as we did for the IMM-based detection scheme. In Figure 5.3 we present example relations between probabilities and threshold for the case when a closed valve is commanded to open (corresponding fault is stuck shut or *failed* in *close* mode).

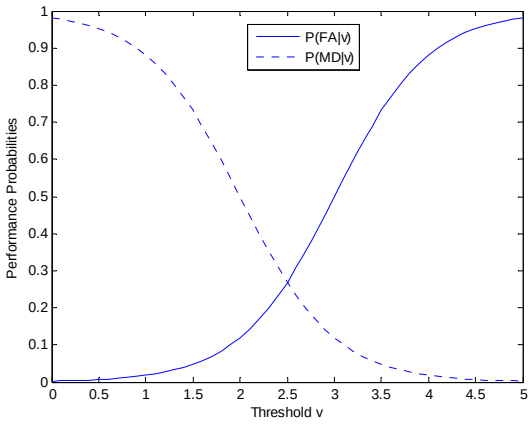

Figure 5.3: Performance probabilities vs. threshold for the knowledge-based detection scheme

The above curves are based on 5 volt fuel pressure sensor readings (horizontal axis) where the fault output is either 0 or 1 indicating the valve as *close* or *open*, respectively. Note that, as we increase the threshold, the probability of false alarm (FA) increases because there is greater chance of sensor reading 0 when it actually might be 1 and hence producing an incorrect detection of the valve as *close* when it actually is *open*. Analogous behavior is exhibited in probability of missed detection (MD). Note that the chosen sigmoid functions are monotonic and given by

$$P(FA \mid v) = \frac{1}{1+e^{-2(v-3)}},$$
$$P(MD \mid v) = \frac{1}{1+e^{2(v-2)}}. \tag{5.4.12}$$

Although the two fault detection schemes presented above are based on different models, they share key properties that can be utilized. Specifically, we can manage their thresholds to resolve conflicts at the expense of decreased performance in terms of MD and FA probabilities. Below we introduce a framework to recursively optimize thresholds so that the conflicts are minimized while maximizing performance in terms of MD and FA probabilities.

5.5 Conflict Resolution

Consider the following conflict resolution cost function J_c.

$$J_c = \sum_i \left[a_-^i P\left(FA^i \mid v^i\right) + a_+^i P\left(MD^i \mid v^i\right)\right] + q\left|b^1\left(v^1, U^1\right) - b^2\left(v^2, U^2\right)\right|$$
$$a_+^i > 0, a_-^i > 0, q > 0 \forall i : i \in \{1,2\} \tag{5.5.1}$$

In this equation, superscript i represents the detection scheme, a_+^i, a_-^i, q are penalty factors or weights, v^1 and v^2 are thresholds, and U^1 and U^2 are inputs to the detection schemes.

Objective function (5.5.1) has two main terms. The first term can be interpreted as a measure of risk incurred by changing the thresholds. The second term penalizes conflicts between the specific faults detected by the two schemes. Note that the second term depends on U^i which represents the command and sensor signals available to the detection scheme i.

5.5.1 Threshold Optimization

Define the first term of (5.5.1) as

$$J = \sum_i \left[a_-^i P\left(FA^i \mid v^i\right) + a_+^i P\left(MD^i \mid v^i\right) \right] \quad i \in \{1,2\} \tag{5.5.2}$$

Note that

$$\frac{\partial J}{\partial v^i} = \frac{\partial J}{\partial P(FA^i)} \frac{\partial P(FA^i)}{\partial v^i} + \frac{\partial J}{\partial P(MD^i)} \frac{\partial P(MD^i)}{\partial v^i} \quad i \in \{1,2\} \tag{5.5.3}$$

We assume nominal threshold values $\bar{v}^i$ are chosen to minimize J so that

$$v^i = \bar{v}^i \Rightarrow \frac{\partial J}{\partial v^i} = 0 \quad \forall i \in \{1,2\} \tag{5.5.4}$$

We denote $J(\bar{v}^1, \bar{v}^2)$ by J^*. Note that J is a separable function i.e. we can optimize FA and MD probabilities for each fault detection scheme separately. The following result then emerges.

Theorem 1: Based on the assumptions in Section 5.3, the minimum of J_c is achieved at thresholds v^1 and v^2 such that either $v^1 = \bar{v}^1$ or $v^2 = \bar{v}^2$ or both.

Proof: The proof follows from the observation that the fault flag changes if there is a change in the threshold exceeding the residual, monotonicity of $P(FA^i|v^i)$, $P(MD^i|v^i)$ as functions of v^i and positivity of the weights.

5.5.2 Residual-based Conflict Resolution

We now present our first method of conflict resolution that uses knowledge of residuals for threshold variation when the two fault detection schemes produce inconsistent decisions regarding the presence of a particular fault. Underlying this method is a

mathematical formulation of the threshold update equations for both fault detection schemes. With threshold updating, we are able to resolve fault decision conflicts in situations where the disagreement is not strong. By strong disagreement, we mean the cases for which the required change in thresholds causes the cost function increase beyond a given bound. We define the upper bound of cost function based the penalty weight on the conflict. For the cases where disagreement is strong, the thresholds are kept at their optimal values while an error signal is generated to alert a higher level supervisor that the conflict was not resolved.

The upper bound on cost function is defined as

$$J_{\max} = J^* + q \tag{5.5.5}$$

The cost of resolving the conflict based on the knowledge of the residual signal for each scheme is given by

$$\begin{aligned} &J^i = J(\bar{v}^i + \Delta v^i, \bar{v}^j) \\ &i, j \in \{1,2\}, i \neq j \end{aligned} \tag{5.5.6}$$

Here, Δv^i is the change in threshold for the i^{th} scheme required to resolve the conflict without changing the threshold for the other scheme.

The threshold for each scheme is updated based on the following equation

$$\begin{aligned} &v^i(t+1) = \begin{cases} \bar{v}^i(t) + \Delta v^i & if: \min\left(J_{\max}, J^1, J^2\right) = J^i \\ & or: J^1 = J^2 < J_{\max}, i = 1 \\ \bar{v}^i(t) & otherwise \end{cases} \\ &i \in \{1,2\} \end{aligned} \tag{5.5.7}$$

5.5.3 Conflict Resolution based on the Markov Decision Process

In this subsection, we present a different solution to the conflict resolution problem based on a Markov Decision Process (MDP) framework. In this case we eliminate assumption

A.2 of Section 5.3 which states that we have knowledge of residual signals. Hence we solve the problem of conflict resolution for any two schemes with independent fault detection such that each fault flag depends upon the value of a scalar parameter. Even though this parameter is not necessarily a threshold, we will refer to it as a threshold for consistency. The optimal MDP policy is then generated for adjusting the threshold(s) of the schemes to reach consensus between the fault flags.

The MDP framework uses a set of MDP states s_1, s_2, s_3… and leads to an optimal policy that can maximize the time-discounted utility of states i.e.

$$Val^* = \arg\max_{\pi} E\left[\sum_{t=0}^{\infty} \gamma^t R(s_t) \mid \pi\right]. \tag{5.5.8}$$

The states of an MDP-based conflict resolution algorithm for a fault detected by two schemes can be defined as:

$$\begin{aligned} &S = \left\{s_1, s_2, s_3, ..., s_N\right\} \\ &where, \\ &s_i = \left\{b_i^1, b_i^2, v_i^1, v_i^2\right\}, \\ &i \in \left\{1, 2, ..., N\right\}. \end{aligned} \tag{5.5.9}$$

Here, S contains states with all possible values of fault flags and thresholds related to the fault. We represent the threshold value set with a finite number of equally-spaced discrete values v_i^k . Number of states N depends upon the size of Δv and the range of v. In particular, if the number of possible values of each threshold is z, then $N = 4z^2$.

The actions are represented as

$$M = \left\{\mu_+^1, \mu_-^1, \mu_+^2, \mu_-^2, NOOP\right\} \tag{5.5.10}$$

and correspond to the following possible actions: 1) increasing (μ_+^1) or decreasing (μ_-^1) the threshold of the first detection scheme by Δv^1; 2) increasing (μ_+^2) or decreasing (μ_-^2) the threshold of the second detection scheme by Δv^2; 3) no change (*NOOP*). Note that each action can result in four possible states. This is because we can change the threshold but cannot guarantee a specific transition in fault flags as the latter is determined by other signals the flags depend upon. Note that NOOP results in states with the same thresholds and same values of fault flags.

Rewards for each state can be represented as

$$R(s_i) = \exp\left(-\phi_+^1 P(MD^1 \mid v_i^1) - \phi_-^1 P(FA^1 \mid v_i^1) - \phi_+^2 P(MD^2 \mid v_i^2) - \phi_-^2 P(FA^2 \mid v_i^2) - \varphi\left|b_i^1 - b_i^2\right|\right)$$
$$i \in \{1,2,...,N\}, \phi_+^k, \phi_-^k, \varphi > 0 \tag{5.5.11}$$

Reward depends upon MD and FA probabilities as well as conflicts in each state.

To find the transition probabilities, statistical information about the fault detectors used should be collected. This can be done by using Monte Carlo simulations, in particular by observing the transitions of the fault flags in response to increment and decrement in each instantiation of the thresholds. From the obtained information, probabilities of fault flag transition can be calculated:

$$\begin{aligned} &T = T(s_i, \mu_r^k, s_p) \\ &k \in \{1,2\}, r \in \{+,-\}, i, p \in \{1,2,...,N\} \end{aligned} \tag{5.5.12}$$

In the MDP, an optimal policy can be calculated using the value iteration algorithm. The policy that selects the optimal action may be found as

$$P^*(s_i) \in \arg\max_{k,r}\left(\sum_{s_p \in S} T(s_i, \mu_r^k, s_p) Val(s_p)\right) \tag{5.5.13}$$

There is a direct relationship between the utility of a state and the utilities 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:

$$Val_{t+1}(s_i) = R(s_i) + \max_{k,r}\left(\sum_{p\neq i}\gamma T(s_i,\mu_r^k,s_p)Val_t(s_p)\right)$$
$$r\in\{+,-\}, k\in\{1,2\} \qquad (5.5.14)$$

where $Val_{t+1}(s_i)$ is the utility of state s_i at iteration $t+1$, $R(s_i)$ is the immediate reward of state s_i, and $T(s_i, \mu_r^k, s_p)$ is the probability of transitioning from state s_i to s_p by executing action μ_r^k. With this structure, the MDP computes the best available threshold setting (action) for each state. The computational complexity of the value iteration algorithm is $5N^2$ or $O(N^2)$ per iteration of Equation (5.5.14). The number of iterations required for convergence within a specified error tolerance depends upon the tolerance itself and the discount factor γ.

5.5.4 The Supervisor Alert

Since neither of our conflict resolution schemes guarantee 100% resolution of conflicts, it is important to have a supervisor (human or software) that can handle strong conflicts. A detailed algorithm for such a supervisor is beyond the scope of this chapter. However, one approach to generating an alert flag for a supervisor can be based on the following equation:

$$e(t) = \prod_{l=t-k+1}^{t}\{c(l)\kappa(l)\}$$
$$c(t) = \left|b^i(v^i(t),U^i) - b^{\bar{i}}(v^{\bar{i}}(t),U^{\bar{i}})\right|, \kappa(t) = \left(c(t-1)c(t)\right)\left|b^i(v^i(t),U^i) - b^{\bar{i}}(v^{\bar{i}}(t-1),U^{\bar{i}})\right|$$
$$\bar{i}, i\in\{1,2\}, i\neq\bar{i} \qquad (5.5.15)$$

Equation (5.5.15) has two interesting properties. First, it has a moving window that indicates a persistent conflict. The moving window is important to avoid intermittent anomalous situations that may be due to short-term external or internal disturbances. The second feature is the use of an oscillation flag that avoids the generation of an error flag for the case of non-persistent fault flags causing a persistent conflict. The oscillation flag can also be used to detect particular failures such as power system failures causing fluctuations in voltages, etc. The information about unresolved faults can be used to make adjustments in system models and/or in the fault detection schemes to account for a change in the environment or the system itself.

5.6 Simulation Results

5.6.1 Residual-based Conflict Resolution

We tested our threshold adjustment conflict resolution strategies for the case study of Section 5.4. Recall that for IMM-based fault detection the residual is the difference between the threshold v^1 and the probability of fault mode, and for logic based fault detection the residual is the difference between the threshold v^2 and the voltage output of the fuel pressure sensor. Based on our assumptions, the fault flag switches when the residual changes sign. For these simulations, we command the thruster *on/off* periodically with time period of 20 time steps and incorporate zero mean Gaussian noise in the residuals with variance of 60% of their maximum value i.e. variance of 3 for the 5 volt sensor output in logic based fault detection and variance of 0.6 for the probability of fault mode in IMM based fault detection. We did not inject the fault in our simulations. Therefore the conflict resolution is equivalent to mitigating a false alarm in one of the two schemes.

We selected the cost function as

$$J = 5P(MD^1 \mid v^1) + 7P(FA^1 \mid v^1) + 5P(MD^2 \mid v^2) + 7P(FA^2 \mid v^2) + \left| b^1(v^1, U^1) - b^2(v^2, U^2) \right| \tag{5.6.1}$$

The value of J^* is 6.3129 and J_{max} = 7.3129 from (5.5.5) and (5.6.1). We measure the performance gain for our conflict resolution scheme as

$$PG = \frac{J_{\max} - J(v^1, v^2)}{J_{\max}} \times 100 \tag{5.6.2}$$

Table 5-1 shows simulation results for the conflict resolution algorithm based on the knowledge of residuals. Note that the conflict is resolved 100% of the time and the average performance gain is about 54%.

Table 5-1: Simulation results for residual-based conflict resolution

No. of Simulations	No. of conflicts incurred	No. of Conflicts Resolved	Average Performance Gain
50	5	5	54.15%
500	52	52	54.%
5000	554	554	54.05%
25,000	2463	2463	54.08%

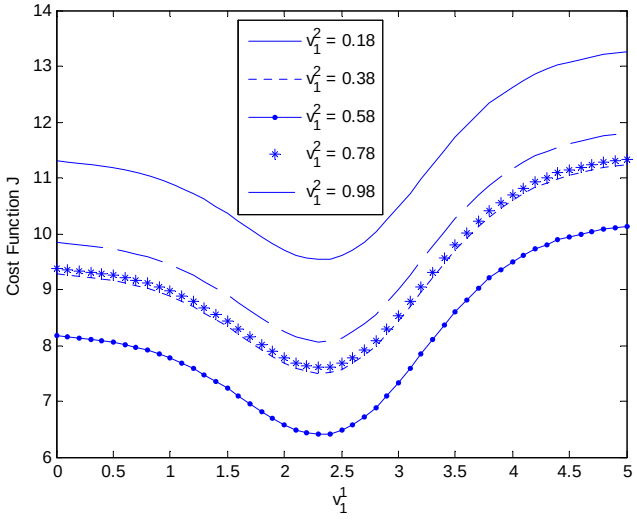

Figure 5.4: Cost as a function of thresholds

5.6.2 MDP Based Conflict Resolution Method

We next consider the use of the MDP conflict resolution algorithm. Also, in these simulations, the thruster is commanded *on* and *off* at the same period as above. We used the following parameter values for our simulations.

$$\Delta v^1 = 0.25, \Delta v^2 = 0.05, \varphi_+^1 = 5, \varphi_-^1 = 7, \varphi_+^2 = 5, \varphi_-^2 = 7, \phi = 100$$
$$z = 21, N = 1764, v^1 \in [0,5], v^2 \in [0,1], \gamma = 0.9, s_{initial} = \{0,0,2.25,0.55\}$$

Our numerical experiment consisted of executing the optimal policy on fault flags generated from both schemes based on similar residual signals as for simulations with the residual-based algorithm. The difference here is that the MDP does not make use of the residual signals and hence the number of optimal actions required to resolve a given conflict is unknown.

Table 5-2: Simulation results for MDP-based conflict resolution with 2 actions per conflict

Simulation Time steps	No. of conflicts incurred	No. of Conflicts Resolved	Average Performance Gain
50	8	6	53.2%
500	41	34	53.1%
5000	413	312	52.97%
25,000	2215	1698	52.99%

Table 5-2 shows the simulation results where we allowed a maximum of two actions per conflict. The performance gain is evaluated based on (5.6.2). Note that the MDP is unable to resolve all the conflicts within 2 changes in the thresholds. But if we allow more changes, more conflicts can be resolved as shown in Table 5-3 where we allowed the MDP to execute up to 5 actions per conflict.

From the comparison of results in Table 5-2 and Table 5-3, we might conclude that if we allow a sufficient number of actions for an MDP based resolution scheme, we can get 100% conflict resolution. It is important to understand, however, that this is not always

the case. For example, when the conflict is strong, the required change in thresholds will be so large that the reward obtained for resolving the conflict will be less than the loss of reward due to use of a sub-optimal threshold. Also, in a real time implementation, there can only be so many updates allowed within each time step depending upon the length of each step and time taken by each update.

Table 5-3: Simulation results for MDP-Based conflict resolution with 5 actions per conflict

Simulations Time Steps	No. of conflicts incurred	No. of Conflicts Resolved	Average Performance Gain
50	3	3	54.04%
500	39	39	54.04%
5000	386	386	54.04%
25,000	2052	2052	54.04%

5.7 Possible Extensions in Proposed Methods

The ideas of conflict resolution presented above can be extended in two possible ways. One way is to allow variable step size in the change in thresholds and the other way is to impose a time constraint on conflict resolution in terms of the number of allowable threshold changes for the resolution of the conflict(s) in the given state. Both of these changes can be exercised simultaneously. The variable step size of the change in thresholds would require additional actions without necessarily requiring a larger state space. On the other hand, the time constraint on the conflict resolution would require an additional variable in the state space to represent the count of the changes in the thresholds exercised for the resolution of the conflict(s) at hand.

The possible advantages of allowing variable changes in the threshold are the reduction in the fluctuation of the thresholds, added precision in the selection of optimal conflict resolving thresholds, and reduction in the number of changes required to resolve the conflict. The fluctuations in the thresholds can be reduced by allowing smaller step sizes

for the threshold change so that the chances of encountering a case where one threshold value is too high and the other threshold value is too low are reduced. On the other hand, reduction in the number of changes required to resolve a conflict is facilitated by large step sizes.

The constraint on the number of changes in thresholds allowed for the resolution of the conflicts can better represent some practical situations where the spacecraft has a hard deadline for the fault detection framework to make a definite fault call. Adding a variable for counting the number of threshold changes also requires modification in the reward function. This is to reflect the importance of resolving the conflict within the allowed number of threshold changes. One could formulate the problem such that if the conflict is not resolved within the allowed attempts of changes in the thresholds, the MDP will transition to a state that will indicate a failure to resolve the conflict. In such situations, maximum likelihood or statistics-based methods can be used to resolve the conflicts that are not resolved by the MDP policy.

5.8 Collaborative Fault Detection

Fault detection and diagnosis has been an area of active research for over four decades. Many fault detection schemes have been proposed that are either based on the dynamics of the system [75][111][112][40] or on a compositional (symbolic) knowledge-based model of the system such as in [75] and [104]. In addition to the development and improvement of specific fault detection and isolation algorithms, integration of these algorithms into an overall diagnostic and fault management (DFM) system is required [70][71]. Such a system must be capable of optimally scheduling an execution monitor (considering priorities and conflicts) to decide if a fault is present, to coordinate between

passive and active diagnostics and control strategies, and to reconfigure the system to a safe/degraded functionality state in the event of fault occurrence in a manner that maximizes system availability. An advanced DFM system is clearly an enabling technology for the autonomous spacecraft and other autonomous aerospace systems.

To improve the quality of fault detection decisions, it is advantageous to have more than one independent detector. But multiple detectors can result in conflicting fault calls. While majority voting can be used in systems with triple or greater redundancy, it does not always work for systems with an even number of fault detectors. Additionally, voting is only robust in cases where data sources and processing algorithms are independent.

Depending upon how fault detection is implemented, there are certain factors that affect the switching of fault flags. Most algorithms require numerical thresholds. In fault detection, a threshold can be defined as an upper or lower bound on the deviation of a signal from its nominal or expected behavior. More generally, a threshold can be any parameter that effects the operation of a fault detector. Almost all fault detection schemes are based on thresholds either explicitly or implicitly. Two fault detectors deployed within the same system, utilizing independent or at least partially independent sources of information, can share fault detection decisions to improve the speed and reliability of fault decisions. This chapter studies how such shared information can improve fault detection decisions, specifically in the context of a system with both compositional and dynamics-based fault detection models.

To facilitate the coexistence of fault detectors, a framework is desired that can not only affect the fault decisions of both detectors, but that can also serve as a platform for

information sharing and improvement of decision quality. The framework needs to take into its account the probabilities of missed detections and false alarms based on the knowledge of the system and the transition probabilities of fault flags given the thresholds of the flags.

The focus of this part is on the development of one element of the full DFM system, specifically a fault detection and diagnosis framework with the ability to resolve conflicts between itself and an external fault detector. In a situation when the fault detectors disagree, the conflict must be resolved to yield a consistent final decision.

In this work, we assume that the external fault detector has fault decision logic that depends on a set of thresholds. We also assume that the thresholds and fault decisions of this external detector are available to our proposed detection and diagnosis framework. We consider the problem of reaching consensus through either adjusting these thresholds in the external detector, or through changing the fault calls made by our detection and diagnosis framework. Note that by changing a threshold, the external detector may or may not change its fault call, depending on the inputs. Consequently, the change in the fault flag can be interpreted as a random event.

To illustrate our approach, we consider attitude control of a single degree-of-freedom (1DOF) spacecraft maneuvered by a battery-powered (electric) reaction wheel system. The external detector in our example is an observer-based fault detector. This detector indicates a fault if the dynamic behavior of the spacecraft is off-nominal. Detection logic for this detector is assumed to be based on two thresholds. Simulation results are presented which highlight the capability of our approach to resolve conflicts.

To mitigate computational complexity, we present an approximate dynamic programming (ADP) approach that can be used to generate near-optimal policies at a reduced computational cost. The proposed ADP approach is based on task-based decomposition of the original state space to create smaller MDPs, solution of smaller MDPs to obtain corresponding optimal value functions, and recombination of these value functions to obtain heuristic estimate for the optimal values of original states.

In the next section, we present our MDP-based collaborative fault detection framework. Sections 5.10 and 5.11 describe ADP decomposition and recombination, respectively, applied to collaborative fault detection. Section 5.12 describes a spacecraft example with simulation results presented in Section 5.13.

5.9 Main Framework

The Markov Decision Process (MDP) or stochastic dynamic programming (SDP) framework outputs optimal decisions for problems involving complex decision making in the presence of uncertainties. An MDP is composed of four basic elements i.e. set of states, set of actions, a state dependent reward function, and action and state dependent transition probabilities. Below we describe each of these four elements for our proposed framework. Figure 5.5 shows interconnection and signal flow between our proposed framework, external observer-based fault detector, and the system to be diagnosed.

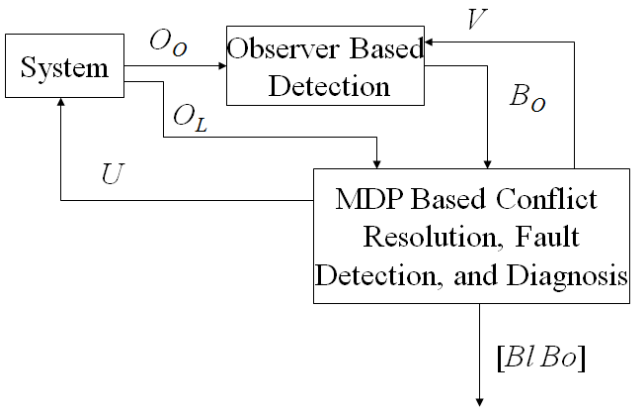

Figure 5.5: Signal flow diagram.

In Figure 5.5, O_o represents the vector of sensor outputs related to the dynamic observer, and O_L represents the vector of observed values of logic-based operating modes of system components. Vector U represents diagnostic or data gathering commands. Vector V represents thresholds used in observer-based fault detection. Vector *Bo* contains binary fault flags for faults detected by the observer-based fault detector. Vector *Bl* contains binary fault flags generated by logic based fault detector built within our framework. Note that in Figure 5.5, thresholds for the observer-based detector are dictated by the MDP based framework so that they can be varied for conflict resolution. Also, *Bo* provides information that is used for both fault detection and conflict resolution in the MDP-based framework. The combined information of all faults [*Bo Bl*] can be used as the final fault decision set if the conflicts are resolved, or for generating alarms if MDP fails to resolve the conflicts in a specified number of time steps. Data gathering commands U are used to obtain updated O_L in response to predefined diagnosis actions. This is useful when the available O_L does not provide enough information to resolve a conflict or to detect a fault.

5.9.1 States

The state space is defined as

$$
\begin{aligned}
&S = \{s_1, s_2, ..., s_N\} \\
&Where, \\
&s_i = \{Bl_i, Bo_i, U_i, O_i, V_i\}: \\
&Bl_i = \{bl_i^1, ..., bl_i^{m_1}\}, Bo_i = \{bo_i^1, ..., bo_i^{m_2}\}, U_i = \{u_i^1, ..., u_i^{m_3}\}, \\
&O_i = \{o_i^1, ..., o_i^{m_4}\}, V_i = \{v_i^1, ..., v_i^{m_5}\}
\end{aligned}
\tag{5.9.1}
$$

A state contains five types of information. In a state s_i, vector Bl_i contains values of m_1 logic-based fault flags, vector Bo_i contains values of m_2 observer-based fault flags, vector U_i contains values of m_3 diagnostic commands, vector O_i contains observed values of m_4 logic-based component modes, and vector V_i contains values of m_5 thresholds for the m_2 observer-based fault flags. Note that vector O in (5.9.1) is same as O_L in Figure 5.5. Subscript L has been removed because there are only logic-based observations in the proposed MDP formulation.

5.9.2 Actions

Actions can be defined using the following equation:

$$
M = \{\mu_{bl1}, ..., \mu_{blm_1}, \mu_{u1}, ..., \mu_{um_3}, \mu_{v1+}, \mu_{v1-}, ..., \mu_{vm_5+}, \mu_{vm_5-}, NOOP\} \tag{5.9.2}
$$

There are m_1 actions of the form μ_{bli} that are used to set the logic based fault flag i (from 1 to 0 and vice versa), m_3 actions of the form μ_{ui} that are used to issue data gathering (or diagnostic) commands, $2m_5$ actions of the form μ_{vi+} and μ_{vi-} that are used to increment and decrement, respectively, the value of threshold i that affects an observer-based fault flag. The amount by which the thresholds are incremented or decremented is assumed to

be pre-specified and fixed. Finally, there is a NOOP (no operation) action to complete the set.

5.9.3 Reward Function

The reward function can be defined using the following equation

$$R(s_i) = \lambda e^{-J(s_i)}, J(s_i) = \left\{ \begin{array}{l} \sum_{k=1}^{m_1} a_k^1 P(FA_k \mid U_i, O_i) + a_k^2 P(MD_k \mid U_i, O_i) \\ + \sum_{k=1}^{m_2} a_k^3 P(FA_k \mid V_i) + a_k^4 P(MD_k \mid V_i) \\ + \sum_{k1,k2 \in Cn} q_{k1,k2} \left| bl_i^{k1} - bo_i^{k2} \right| - H(U_i, O_i, Bl_i, Bo_i) \end{array} \right\} \quad (5.9.3)$$

In (5.9.3), a_i^j and $q_{i,j}$ for all i, j are constants. $P(FA_j|U_i, O_i)$ and $P(MD_j|U_i, O_i)$ represent probabilities of false alarm and missed detection, respectively, in state i for the fault represented by flag bl_j, given diagnostic commands and observations in state i. $P(FA_j|V_i)$ and $P(MD_j|V_i)$ represent probabilities of false alarm and missed detection, respectively, in state i for fault represented by flag bo_j, given threshold values in state i. H quantifies the amount of information available in each state. The reward function is an inverse exponential of a cost function which is a sum of four distinct terms. The first term represents the weighted sum of the probabilities of false alarms (*FA*) and missed detections (*MD*) for the logic-based fault detector. These probabilities are assumed to be dependent upon diagnostic inputs and observations. We assume that the corresponding conditional distribution can be represented in the form of a Bayes Net so that a Bayesian inference algorithm can be used to compute these probabilities for each state. The second term in the cost function represents the weighted sum of the probabilities of false alarm and missed detection for the observer-based fault detector. Here, we assume that these

probabilities have been computed using Monte Carlo simulations or other similar statistical methods. The third term in the cost function represents penalty weights on the possible conflicts. Here, *Cn* is the set of faults which are detected by both detectors. Each member of the set *Cn* is a pair (*k1*, *k2*) where *k*1 is the index of the fault in *Bl* and *k*2 is the index of the same fault in *Bo*. The fourth term in the cost function quantifies the amount of information available about the system. This term is negative because more information improves the solution thus reduces cost.

5.9.4 Transition Probabilities

When an action μ_k is executed from a state s_i, the transition probability $T(s_i, \mu_k, s_j)$ indicates the probability of transition to state s_j. In our formulation, there are three types of actions (besides NOOP). The first type consists of deterministic actions that switch the logic-based fault flags. Transition probabilities for these actions can be represented as

$$T\left(s_i, \mu_{blk}, s_j\right) = \begin{cases} 1 & if : \left(s_i \setminus bl_i^k = s_j \setminus bl_j^k\right) \wedge \left(bl_j^k = 1 - bl_i^k\right) \\ 0 & otherwise \end{cases} \qquad (5.9.4)$$
$$k \in \{1, 2, ..., m_1\}, i, j \in \{1, 2, ..., N\}$$

We use the \ operator to indicate set difference. The second type of actions are diagnostic command actions which are also deterministic since a command can be either issued ($u_j^k = 1$) or not issued ($u_j^k = 0$).

$$T\left(s_i, \mu_{uk}, s_j\right) = \begin{cases} 1 & if : \left(s_i \setminus u_i^k = s_j \setminus u_j^k\right) \wedge \left(u_j^k = 1\right) \\ 0 & otherwise \end{cases} \qquad (5.9.5)$$
$$k \in \{1, 2, ..., m_3\}, i, j \in \{1, 2, ..., N\}$$

Although Equation (5.9.5) indicates deterministic transitions, there may be some transitions that are not directly caused by actions. For example, once a diagnostic observation is received, the state changes. This change is not deterministic because observations are affected by random noise and other factors. The third type of action is threshold changes (or conflict resolution actions). These actions change the thresholds deterministically but the changes in corresponding observer-based fault flags are represented as random events.

Equation (5.9.6) is based on the assumption that each threshold cannot affect more than one observer-based fault flag whereas each flag may be affected by multiple thresholds (i.e. $m_5 \geq m_2$). Equation (5.9.6) also represents actions to increase thresholds. A similar equation can be used to define actions that decrease thresholds. Variable p is the probability of fault flag switching and is dependent upon values of threshold and the corresponding change Δ in the threshold. In general, p could be a function of time and control inputs, but here we assume p to be a function of thresholds only. Finally, $\bar{v}^k$ represents the upper bound on the value of k^{th} threshold and $\hat{k}$ represents the index of the fault flag affected by threshold k.

$$T\left(s_i,\mu_{vk+},s_j\right)=\begin{cases} p & if:\left(s_i\setminus v_i^k=s_j\setminus v_j^k\right)\wedge\left\{\begin{matrix}\left(v_j^k=\Delta^k+v_i^k\right)\\ \vee\left(v_j^k=v_j^k=\overline{v}^k\right)\end{matrix}\right\} \\ & \wedge\left(bo_j^{\hat{k}}=1-bo_i^{\hat{k}}\right) \\ 1-p & if:\left(s_i\setminus v_i^k=s_j\setminus v_j^k\right)\wedge\left\{\begin{matrix}\left(v_j^k=\Delta^k+v_i^k\right)\\ \vee\left(v_j^k=v_j^k=\overline{v}^k\right)\end{matrix}\right\} \\ & \wedge\left(bo_j^{\hat{k}}=bo_i^{\hat{k}}\right) \\ 0 & otherwise \end{cases} \tag{5.9.6}$$

$$k\in\left\{1,2,...,m_5\right\},i,j\in\left\{1,2,...,N\right\},\hat{k}\in\left\{1,2,...,m_2\right\}$$

5.9.5 Solving the MDP: Value Iteration

There are three major ways of solving MDPs: Policy Iteration, Value Iteration, and Linear Programming. All these methods calculate stationary optimal policies over an infinite horizon. We use value iteration as presented in Chapter 2. The MDP can also be solved using a backward induction algorithm [84] for obtaining a finite-horizon time-varying optimal policy.

5.10 ADP Decomposition Approach

Section 5.9 provides an optimal decision making framework that is computationally intensive. This section proposes a strategy to mitigate complexity by decomposing the full MDP into three sub-problems (MDP 1 – MDP 3), each of which is described below.

5.10.1 MDP 1: Logic Based Fault Detection

This MDP is dedicated to the switching of logic-based fault flags based on information about diagnostic inputs and observations along with the information of fault flags from an observer-based detector.

States—The set of states is defined as:

$$\begin{aligned} &S1 = \{s_1, s_2, ..., s_{N1}\} \\ &\textit{where each state is represented as,} \\ &s_i = \{Bl_i, Bo_i, U_i, O_i\}: \\ &Bl_i = \{bl_i^1, ..., bl_i^{m_1}\}, Bo_i = \{bo_i^1, ..., bo_i^{m_2}\}, \\ &U_i = \{u_i^1, ..., u_i^{m_3}\}, O_i = \{o_i^1, ..., o_i^{m_4}\}. \end{aligned} \tag{5.10.1}$$

Only the last component of the state in Equation (5.9.1) has been dropped in (5.10.1). However, this nontrivially reduces MDP complexity since m_5 is typically large. Also values in *Bo* do not change in *MDP1*.

Actions—The set of actions is defined as:

$$M1 = \{\mu_{bl1}, ..., \mu_{blm_1}, NOOP\}. \tag{5.10.2}$$

The only actions here are those of switching logic-based fault flags (refer to (5.9.2)).

Reward Function—The reward function is defined as:

$$R1(s_i) = \lambda_1 e^{-J1(s_i)}, J1(s_i) = \left\{ \begin{array}{c} \sum_{k=1}^{m_1} a_k^1 P(FA_k \mid U_i, O_i) + a_k^2 P(MD_k \mid U_i, O_i) \\ + \sum_{k1,k2 \in Cn} q_{k1,k2} \left| bl_i^{k1} - bo_i^{k2} \right| \end{array} \right\}. \tag{5.10.3}$$

This reward function is similar to (5.9.3) except for the absence of terms related to information and performance probabilities for the observer-based detector. Note that we use the term performance probabilities to represent the probabilities of false alarms and missed detections.

Transition Probabilities—Transition probabilities can be represented using Equation (5.9.4).

5.10.2 MDP 2: Conflict Resolution

This MDP is dedicated to the resolution of conflicts by adjusting the thresholds for the observer-based detector. The specifics are as follows.

States—States can be defined as:

$$\begin{aligned} &S2 = \{s_1, s_2, ..., s_{N2}\} \\ &Where, \\ &s_i = \{Bl_i, Bo_i, V_i\}: \\ &Bl_i = \{bl_i^1, ..., bl_i^{m_1}\}, Bo_i = \{bo_i^1, ..., bo_i^{m_2}\}, V_i = \{v_i^1, ..., v_i^{m_5}\} \end{aligned} \tag{5.10.4}$$

Diagnostic inputs and observations are not required because we assume there is already a separate observer-based detector in the system that generates *Bo* using any input and/or observation data that it needs along with the model of the dynamics of the system. Also, analogous to MDP1, in MDP2, flag values for *Bl* do not change.

Actions—Actions are the following subset of actions from (5.9.2):

$$M2 = \{\mu_{v1+}, \mu_{v1-}, ..., \mu_{vm_5+}, \mu_{vm_5-}, NOOP\} \tag{5.10.5}$$

Reward Function—The reward function can be defined as:

$$R2(s_i) = \lambda_2 e^{-J2(s_i)}, J2(s_i) = \left\{ \begin{array}{c} \sum_{k=1}^{m_2} a_k^4 P(FA_k \mid V_i) + a_k^4 P(MD_k \mid V_i) \\ + \sum_{k1,k2 \in Cn} q_{k1,k2} \left| bl_i^{k1} - bo_i^{k2} \right| \end{array} \right\} \tag{5.10.6}$$

This reward function is similar to (5.9.3) except for the absence of terms related to information and performance probabilities for the flags in *Bl*.

Transition Probabilities—Transition Probabilities can be defined using (5.9.6).

5.10.3 MDP 3: Information Gathering/Diagnostics

This MDP is dedicated to the task of information gathering or diagnostics.

States—States can be defined using (5.10.1). The set of states in MDP3 are the same as those in MDP1.

Actions— Actions are the following subset of actions from (5.9.2):

$$M3 = \left\{ \mu_{u1}, ..., \mu_{um_3}, NOOP \right\} \tag{5.10.7}$$

Reward Function—The reward function can be defined as:

$$\begin{array}{l} R3(s_i) = \lambda_3 e^{-J3(s_i)} \\ J3(s_i) = -H(U_i, O_i, Bl_i, Bo_i) \end{array} \tag{5.10.8}$$

Transition Probabilities—Transition Probabilities can be defined using (5.9.5).

5.10.4 Integration

We integrate the three MDPs as shown in Figure 5.6, and refer to this as the split MDP solution approach. Recall that O_L in Figure 5.6 corresponds to O in the original MDP

formulation. Also, some of the information generated by one MDP is used by the others e.g. vector *Bl* generated by MDP1 is used by MDP2 and MDP3 etc.

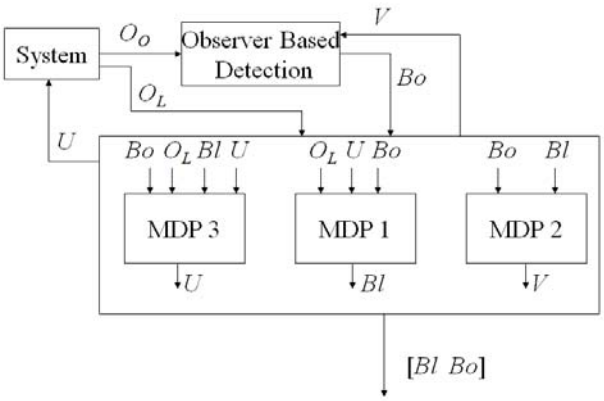

Figure 5.6: Signal flow with the split MDP framework.

5.11 ADP Recombination Algorithm

In this section we present a recombination algorithm that can be used to generate a policy for the states in the framework of Section 5.9 by integrating policies generated by the three decomposed MDPs in Section 5.10. Figure 5.7 shows the recombination steps.

Step 1. Solve each of the smaller MDP's and compute optimal values for all states.

Step 2. Merge the values calculated in step 1 to estimate the optimal values for the recombined MDP as follows:

$$Val^{est}(s) = Val1^{*}(s1) + Val2^{*}(s2) + Val3^{*}(s3)$$
$$where,$$
$$s \in S, s1 \in S1, s2 \in S2, s3 \in S1,$$
$$(s1 = s3) \wedge (Bo_{s2} = Bo_{s1}) \wedge (Bl_{s2} = Bl_{s1}) \wedge (V_{s2} = V_{s})$$

Step3. Calculate the ADP estimate of an optimal policy using the equation:

$$Pol^{est}(s) = \max_{\mu \in M} \left\{ \gamma \sum_{s' \in S} T(s, \mu, s') Val^{est}(s') \right\}$$

for all $s \in S$

Figure 5.7: ADP recombination algorithm for MDP-based framework.

5.12 Implementation Example

To illustrate our framework, we consider a spacecraft reaction wheel fault management example. The reaction wheel (*RW*) is connected to two electronics driver boards (*EB*1 and *EB*2) with corresponding mode monitors (O_{EB1} and O_{EB2}), and a battery, as shown in Figure 5.8. We consider detection of six logic-based faults: a fault in the battery, faults in each of the two electronics boards, faults in each of two monitors for the electronics boards, and a fault in the reaction wheel. We also include an observer-detected fault in the reaction wheel (anomalous torque) with detection logic that depends upon two thresholds v_1 and v_2. A Bayes net that represents the joint distribution of the operational

modes (normal/failed) of the components is shown in Figure 5.9. The Bayes net also includes fault flag b_{RW} and its corresponding thresholds v_1 and v_2. Variables $SW1$ and $SW2$ correspond to the activation switches for the electronics boards. Initial conditional distributions corresponding to the Bayes Net are shown in Table 5-4 for fixed values of thresholds v_1 and v_2. The symbol "-" before a variable in Table 5-4 indicates *faulty* mode except for $SW1$ and $SW2$ where it indicates that the corresponding switch is turned *off*. Note that all probabilities in Table 5-4 are probabilities of components being in *normal* mode given available evidence. The corresponding probabilities of components being in *faulty* modes are calculated by subtracting the probability of the component fault from 1. Using these distributions, the probabilities of failure (given any evidence) for any of the components can be calculated using Bayesian inference.

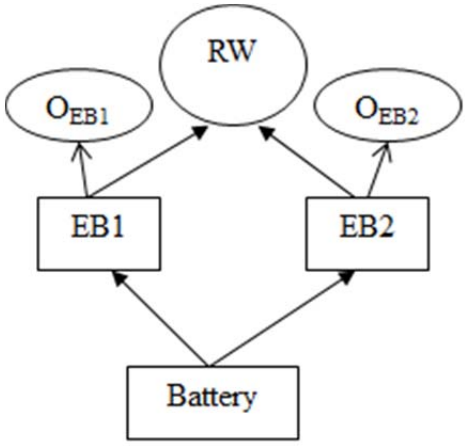

Figure 5.8: Simulation example system.

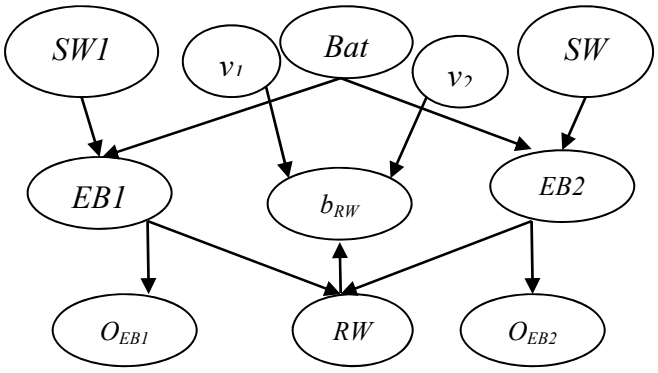

Figure 5.9: Bayes net for the simulation example.

Table 5-4: Conditional probabilities ($v_1 = 0.5$, $v_2 = 0.5$)

P(B)	0.995
P(SW1)	0.5
P(SW2)	0.5
P(EB1|B,SW1)	0.998
P(EB1|-SW1,B)	0.999
P(EB1|SW1,-B)	0.4
P(EB1|-B,-SW1)	0.7
P(EB2|B,SW2)	0.998
P(EB2|-SW2,B)	0.999
P(EB2|SW2,-B)	0.4
P(EB2|-B,-SW2)	0.7
P(O_{EB1}|EB1)	0.95
P(O_{EB1}|-EB1)	0.15
P(O_{EB2}|EB2)	0.95
P(O_{EB2}|-EB2)	0.15
P(RW|EB1,EB2)	0.999
P(RW|EB1,-EB2)	0.999
P(RW|-EB1,EB2)	0.999
P(RW|-EB1,-EB2)	0.2
P(b_{RW}|RW,v_1,v_2)	*P(FA|v_1,v_2)*
P(b_{RW}|-RW,v_1,v_2)	*1-P(MD| v_1,v_2)*

The dynamics of the system are represented by:

$$\begin{aligned} \dot{\theta} &= \omega \\ I\dot{\omega} &= M^{RW} + d + u_f \\ \dot{H}^{RW} &= -M^{RW} \end{aligned} \tag{5.12.1}$$

In (5.12.1), d is assumed to be a zero mean Gaussian disturbance with variance 10^{-4}. Also, θ represents angular displacement of the spacecraft in the inertial frame, ω represents angular velocity of the spacecraft represented in a body fixed frame, whereas, H^{RW} represents angular momentum of the reaction wheel in the spacecraft's body fixed frame. We assume full state feedback with perfect sensors.

For the detection of faults, we used a scheme based on the comparison of nominal and actual dynamics. The nominal dynamics can be represented by similar equation as (5.12.1) except for the disturbance d and fault input u_f that would be zero in the nominal model, specifically:

$$r = \begin{bmatrix} |\theta - \theta_n| \\ |\omega - \omega_n| \end{bmatrix}, V = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}$$
$$b_{RW} = \begin{cases} 1 & if: v_1 - r_1 < 0, or, v_2 - r_2 < 0 \\ 0 & otherwise \end{cases} \tag{5.12.2}$$

Here, b_{RW} is the fault where the reaction wheel turns off and its angular velocity decays to zero at a constant rate, i.e.

$$u_f \in \left\{0, -M^{RW} + \tau H^{RW}\right\}, \tau > 0 \tag{5.12.3}$$

It is easy to show that r converges to $[0\ 0]^T$ whenever $M^{RW} = K(x_{des} - x)$ is selected such that the closed loop system has eigenvalues with negative real parts since there are no disturbances or faults.

To implement our full MDP framework on the spacecraft reaction wheel system, we define the following states:

$$
\begin{aligned}
&S = \{s_1, s_2, ..., s_N\} \\
&s_i = \{Bl_i, bo, O_i, U_i, V_i\} \\
&Where, \\
&Bl_i = \{bl_i^1, bl_i^2, bl_i^3, bl_i^4, bl_i^5, bl_i^6\} \\
&O_i = \{O_i^1, O_i^2\} \\
&O_i^1, O_i^2 \in \{normal, failed, unknown\} \\
&U_i \in \{1,2\}, V_i = \begin{bmatrix} v_i^1 & v_i^2 \end{bmatrix}^T, v_i^k \in [0:0.1:1], k \in \{1,2\}
\end{aligned}
\tag{5.12.4}
$$

In (5.12.4), *Bl* includes 6 fault flags for faults in components shown in Figure 5.8. *O* includes monitored status of the two electronics boards. *U* is a scalar where *U* = 1 indicates that *SW*1 is turned on and *SW*2 is turned off whereas *U* = 2 indicates that *SW*1 is turned off and *SW*2 is turned on. *V* includes the two thresholds that effect observer-based fault flag *bo*. For this domain the total number of states in *S* is 185,856 which is less than all possible combinations of the values of state-components i.e. $2^6 \times 2 \times 3^2 \times 2 \times 11^2 =$ 278,784. This is due to the constraint that if a circuit board is in use, its status cannot be *unknown*. Switching between electronics boards can yield additional information about component failures in cases of anomalous spacecraft behaviors when status of the board not in use is *unknown*. For example, in an anomalous situation, the MDP can command a switch between the electronics boards in order to determine if both electronics boards are providing the same functionality. If changing the board does not change the spacecraft condition, the reaction wheel is more likely faulty. If, on the other hand, changing the board restores nominal or improved operation, then there is an increased likelihood of failure in the original board.

The actions for this domain can be represented as

$$M = \begin{Bmatrix} \mu_{v1+}, \mu_{v1-}, \mu_{v2+}, \mu_{v2-}, \mu_{bl1}, \mu_{bl2}, \mu_{bl3}, \ldots \\ , \mu_{bl4}, \mu_{bl5}\mu_{bl6}, \mu_{SW1}, \mu_{SW2}, NOOP \end{Bmatrix} \tag{5.12.5}$$

All actions in (5.12.5) are similar to the ones defined in (5.9.2) except for μ_{SW1} and μ_{SW2} that correspond to tuning $U = 1$ and $U = 2$, respectively.

The reward function can be obtained using Equation (5.9.3) where we define H as follows:

$$H(Bl_i, bo_i, O_i, U_i) = \begin{cases} h_1 \left| bl_i^1 - bo_i \right| & if: O_{EB(-U_i)} = unknown \\ +h_2 \left(\neg O_{EB(U_i)} \right) & \\ 0 & otherwise \end{cases} \tag{5.12.6}$$

In (5.12.6), bl_i^1 is the logic-based fault flag for a fault in the reaction wheel. This information has positive reward if the status of the electronics board not in use is *unknown*. The value of the reward depends on if there is a conflict between the two detectors, if the in-use electronics board is ruled out as faulty, or both.

The parameter values used for the reward function, adapted from Equation (5.9.3), are

$$\begin{aligned} & a_k^1 = 0.5, a_k^2 = 0.7, \forall k \in \{1, 2, ..., 6\} \\ & a_3 = 0.5, a_4 = 0.7 \\ & h_1 = 0.2, h_2 = 0.3, \lambda = \lambda_1 = \lambda_2 = \lambda_3 = 10^5 \\ & q = 5 \end{aligned} \tag{5.12.7}$$

Note that the parameter q (the penalty weighting parameter for the conflict) has no subscript because there is only one possible conflict. The transition probabilities were specified using Equations (5.9.4), (5.9.5), and (5.9.6). We selected p in Equation (5.9.6) to be a piecewise constant function of *bo* and changes in v_1 and v_2. For $bo = 0$ and an increment in v_1 or v_2, $p = 0$; for $bo = 1$ and an increment in v_1 or v_2, $p = 0.1$; for $bo = 0$

and a decrement in v_1 or v_2, $p = 0.1$; for $bo = 1$ and a decrement in v_1 or v_2, $p = 0$; for no changes in v_1 and v_2 i.e. when the intended increment or decrement is not possible because thresholds are already on their boundary values, $p = 0.02$. To implement the decomposed MDPs on the reaction wheel system, equations from Section 5.10 can be used with the same information used in the integrated MDP model.

Remark on Computational Complexity: The decomposed MDP1 and MDP3 each have 1536 states, and MDP 2 has 484 states. The full MDP had 185,856 states, illustrating the ability of decomposition to enable computational savings. Recall that the computational complexity of value iteration is proportional to the square of the size of the state space times the size of the action space which means that the original MDP has computational complexity $O(10^{11})$ whereas the three split MDPs have combined computational complexity $O(10^6)$. However, this computational savings comes at the cost of loss in performance. Examples of this loss are presented in the next section.

5.13 Simulation Results

In this section, we explain two simulation cases and report the results. Most of the results are generated by the comparison between main, split, and recombined MDP formulations.

5.13.1 Simulation Setup 1

Using the example described in Section 5.12, we implemented the main MDP, the split MDPs, and the ADP-based recombined MDP using value iteration and ADP algorithms described previously. The simulation setup for the main MDP and the ADP-based solution are given in Figure 5.10. As shown in the figure, in the simulation we feed the MDP policy current state per (5.12.4). The MDP policy then executes the optimal action

and state is partially updated. Then we use random number generators to set monitored modes for the electronics boards (O_{EB1} and O_{EB2}) and residuals for the observer-based detector (r_1 and r_2). The state is updated using this new information and is fed into the MDP policy to complete the cycle of one time step.

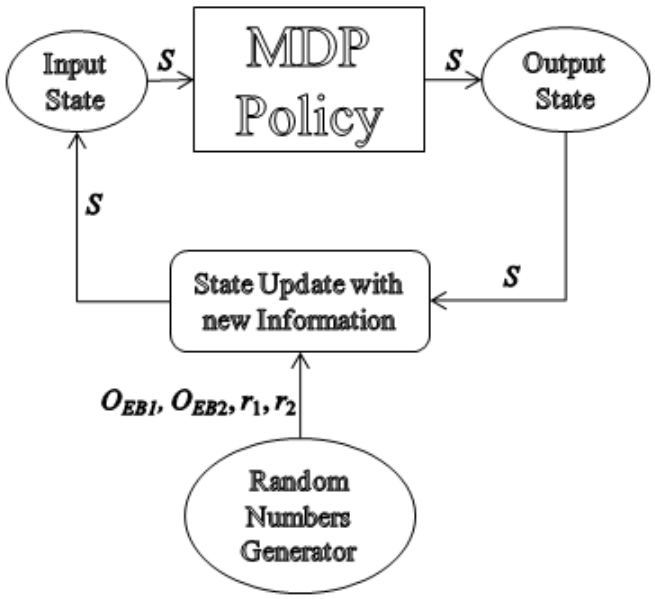

Figure 5.10: Setup for main and ADP-based MDPs.

For the split MDPs, there were quite a few options for implementing a series of combinations. Figure 5.11 shows the particular combination that we used. The cycle of MDP execution begins with the conflict resolution MDP which executes an optimal action and causes changes in some parameters of the state *S*. Note that only a subset of the information in *S* is required to determine the optimal action for the conflict resolution MDP policy (see (5.9.1) and (5.10.4)). The state is updated after the optimal action is complete. The same procedure is repeated for the logic-based fault detection MDP and diagnostic MDP. Here again, only part of the information in *S* is required to determine optimal policies for logic-based detection and diagnostic MDPs. The final updated state

is transferred to the output block after which external updates similar to those in Figure 5.10 are performed before the updated state is fed back into the split MDP block and the cycle is completed.

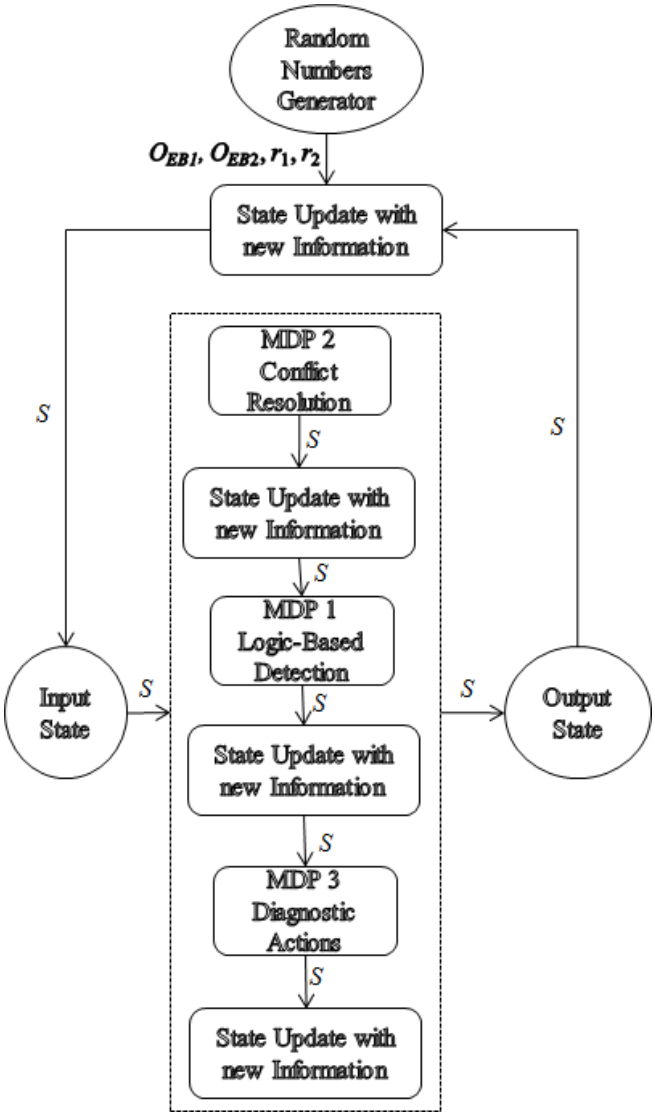

Figure 5.11: Evoluation of System State *S* for split MDPs.

For case 1, false alarm and missed detection probabilities were generated by a similar function as in [71]. For the next case, these probabilities were generated using Monte Carlo simulations on the dynamic model and fault detection scheme presented in

Equations (5.12.1) and (5.12.2) respectively. In the results for case 1, we present percentages of conflicts resolved for two cases with difference in the value of parameter q.

5.13.2 Simulation Results for Setup 1

Case 1 (q = 5)—Figure 5.12 shows the results from simulation setup 1. The trajectories were recorded for 25, 250, and 2500 time steps where each time step corresponds to a complete cycle per Figure 5.10 and Figure 5.11.

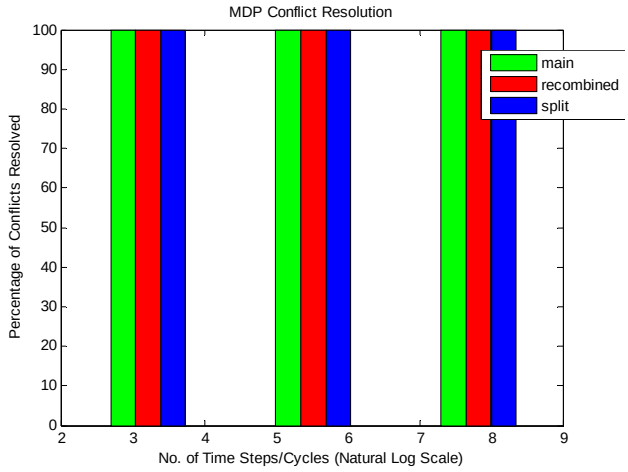

Figure 5.12: Percentage of conflicts resolved by MDPs in setup 1 (with q = 5).

Our result indicates that all conflicts were resolved by all MDPs. This is due to the high penalty factor (q = 5) on conflicts in the reward function of the MDPs. Note that, although resolution of conflicts is a desirable property, this alone cannot determine the optimality of the MDP policy since the reward function is composed of four factors (see (5.9.3)) among which resolution of conflicts is one.

Case 2 (q = 0.5)—Figure 5.13 presents the results for this case.

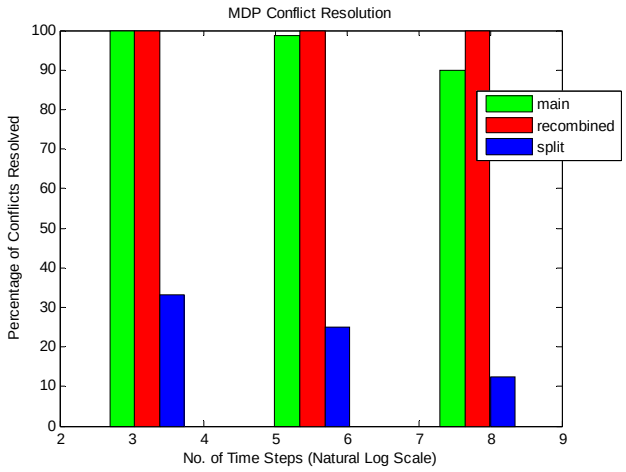

Figure 5.13: Percentage of conflicts resolved by MDPs in setup 1 (with q = 0.5).

Note that the percentage of conflicts resolved is dropped significantly for the split MDP whereas it has also dropped slightly in main MDP. This points to the high sensitivity of the split MDP to the change in q.

5.13.3 Simulation Setup 2

In setup 2, we used the dynamics model (5.12.1) and detection scheme (5.12.2) to generate residuals r_1 and r_2 instead of a random number generator in Figure 5.10 and Figure 5.11. Parameter values used in this setup were from (5.12.7) with q = 5. We carried out four experiments with rest-to-rest maneuvers of the 1DOF spacecraft slewing from 0 to π radians. We selected control gain vector $K = [1\ \ 1]^T$ and τ = 2 for all experiments. The fault in all four experiments corresponds to a nonzero value of u_f in (5.12.3). The initial state for all MDPs in all experiments was the state where all fault flags were turned off. Monitor O_{EB1} of the circuit board $EB1$ showed *normal* as its status whereas the monitor O_{EB2} for the circuit board $EB2$ showed *unknown* as its status. The switch of circuit board $EB1$ was set to 1 whereas the switch of board 2 was set to 0, and the thresholds (v_1, v_2) were set to be (0.6, 0.1). Below, we present a brief description of

the behavior of the dynamics and MDP for each experiment. We also present plots of the dynamics and MDP for our fourth experiment as an example. A performance comparison of the ADP-based approach with the main MDP is shown at the end.

5.13.4 Simulation Results for Setup 2

Experiment 1—In our first experiment, to assess the nominal behavior of the dynamics and MDPs, we did not introduce the fault. In this case, the nominal dynamics and the actual dynamics differ only slightly. This difference is due to disturbance d injected into the simulated system. MDP policies for the main and recombined MDPs yield constant trajectories whereas the split MDP yields some variations in thresholds. This was a conflict-preventive behavior since the logic-based fault flag for the reaction wheel was set to 0. Recall that in the split MDP the logic-based fault flags are fixed from the point of view of the conflict resolution MDP. The main and split MDPs performed a diagnostic action, and once information about EB2 was obtained, no further diagnostic action was performed.

Experiment 2—In the second experiment, we introduced the fault at $t = 1$ *sec*. The resulting actual dynamics showed a significant deviation from the actual dynamics. On the other hand, all the MDPs resolved the conflict by switching the logic-based fault flag for the reaction wheel to 1 after switching of *bo* to 1 at $t = 1$ *sec* due to crossing of thresholds by residuals r_1 and r_2. In this experiment, the recombined MDP made valiant efforts to switch the flag *bo* back to zero but did not succeed since the residuals were very high due to significant deviation of the dynamics from their nominal behavior.

Experiment 3—In our third experiment, we introduce the same fault but at $t = 13$ *sec*. In this case the fault is almost undetectable by the observer since the satellite has almost completed its maneuver. For this case, the MDP behaviors are similar to the no-fault case.

Experiment 4—In our fourth experiment, we introduced the same fault at $t = 1$ *sec* and then we removed the fault at $t = 2$ *sec*. The resulting dynamics are shown in Figure 5.14 where, $x1$ represents θ, $x2$ represents ω, and $x3$ represents H^{RW}. All the MDPs resolved the conflict by switching the logic-based fault flag for the reaction wheel to 1 as before, but after the fault was removed this flag was also switched back to zero.

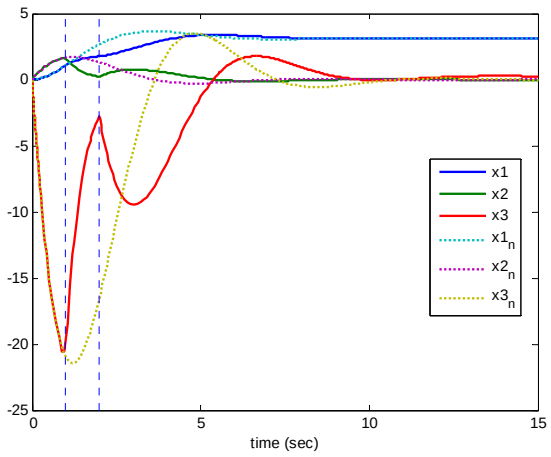

Figure 5.14: Dynamics for the reaction wheel fault at $t = 1$ sec and recovery initiation at $t = 2$ sec.

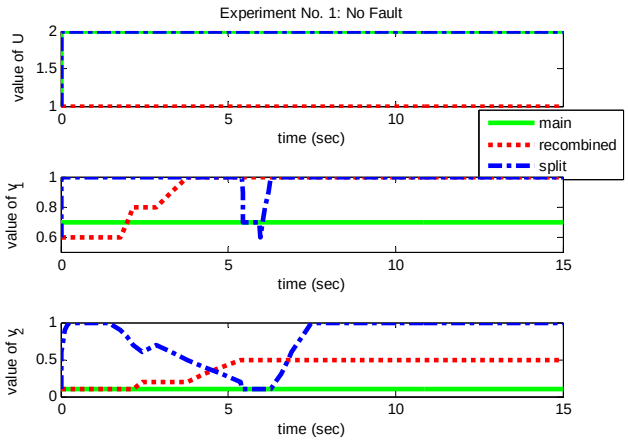

Figure 5.15: Behavior of MDPs for fault at t = 1 sec and recovery at t = 2 sec case.

Figure 5.15 shows the behavior of the diagnostic input U, and the thresholds v_1 and v_2 for MDPs in this experiment. For this case the split and recombined MDPs both performed aggressive threshold changes. This behavior is due to our selection of parameters (5.12.7) where penalty on conflict is ten times larger than the penalty on probabilities of false alarms and missed detections that are based on the threshold values. On the other hand, the main MDP did not try to change the thresholds since the representation of fault flag transition probabilities and hence possibility of conflict generation is accurately represented in the main MDP as opposed to the split and recombined MDPs, where the conflict resolution MDP assumes fixed Bl whereas the fault detection MDP assumes fixed Bo.

Performance of ADP Algorithm—Finally, we present a performance comparison of the main, split, and recombined MDPs (Figure 5.16) based on total expected reward given by Equation (5.13.1).

$$ER = \sum_t \gamma^t T\left(s_{t-1}, \mu_{t-1}, s_t\right) R\left(s_t\right) \tag{5.13.1}$$

Figure 5.16 shows that the recombined MDP performs well in experiments 1, 3, and 4, but poorly in experiment 2. Overall, considering the computational savings (of the order of 10^5), our proposed ADP based approach appears promising.

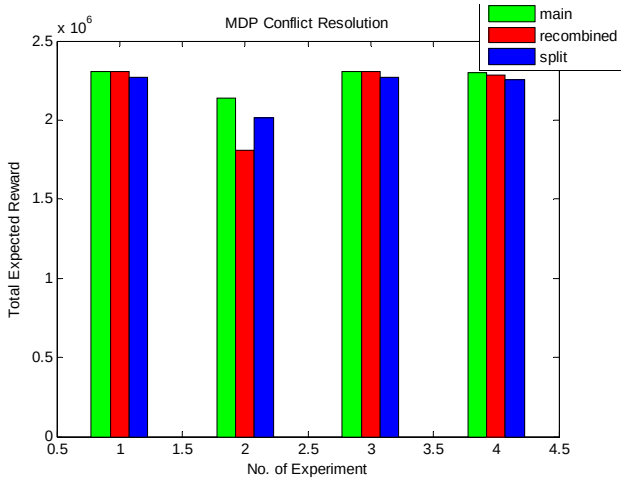

Figure 5.16: Performance comparison for MDPs in all 4 cases.

5.14 Further Analysis of the ADP-based Policy

In this section, we present some results that are analogous to the results presented in Section 4.6.3 in Chapter 4 regarding the mission planning MDP. We use the original and ADP-based MDP formulations in the fault detection case study presented above and simulate the trajectories over finite horizon lengths using the policies calculated for both the original and the ADP-based MDPs. The results of changes in the epoch, the discount factor, the weights of the performance (a_i^j's in Equation (5.9.3)), and the weights on the conflict (($q_{k1,\,k2}$'s in Equation (5.9.3))) are shown in Table 5-5.

Table 5-5: Performance results for the ADP-based MDP policy

Variable Factor	Constant Factors	Best Average performance	Worst Average Performance	Worst case Performance
Epoch: 3-10	$\gamma = 0.9$ Performance Weights = 0.5 Conflict Weights = 5	98.9% for Epoch = 3	96.4% for Epoch = 10	67.8% for Epoch = 3
γ: 0.29:0.1:0.99	Epoch = 3 Performance Weights = 0.5 Conflict Weights = 5	99.8% for γ = 0.29	98.9% for γ = 0.99	70.3% for γ = 0.99
Performance Weights: 1-5 (Conflict Weights = 6 – Performance Weights)	$\gamma = 0.5$ Epoch = 3	99.52% for Performance Weights = 5	99.45% for Performance Weights = 1	68.3% for Performance Weights = 5

5.15 Conclusions and Future Work

In the first part of this chapter, we presented two conflict resolution algorithms capable of resolving differences in faults detected by diverse fault detection schemes. Both our conflict resolution schemes share a common resolution strategy: they change missed detection vs. false alarm thresholds for one or both fault detection algorithms as a method of converging on a common fault set. In the first conflict resolution method, we optimized the changes in thresholds with respect to a cost function that takes into account not only the conflicts but also the probabilities of missed detection and false alarms for both schemes and that uses residuals to update threshold values. In our second strategy, we optimize the change in thresholds using a Markov Decision Process based on rewards, transition probabilities of fault flags, and a discount factor, but without knowledge of residuals. We have demonstrated the ability of both conflict resolution algorithms to

resolve conflicts using a simple example of spacecraft thruster failure. Simulation results show that our approaches are able to resolve conflicts, with the residuals method offering a faster solution and the MDP method offering a more general resolution method not dependent on knowledge of residuals. In future work we plan to extend our models to accommodate more than two fault detection schemes and to handle multiple faults, initially with independent thresholds then ultimately with interdependencies.

In the second part of this chapter, we have presented a Markov Decision Process framework for detecting faults, executing diagnostic actions, and resolving conflicts. We presented task-based decomposition and recombination approaches, with recombination cast as approximate dynamic programming to reduce the computational complexity. To compare the three MDP formulations, we demonstrated a 1DOF spacecraft case study in which the spacecraft must diagnose reaction wheel system faults. Simulations indicate the conflict is always resolved when penalty weight is sufficiently high, but this is not necessarily the case with low penalty weight. Failure to resolve the conflict typically happens when the dynamics-based observer is in strong disagreement with the compositional model. Accurate resolution of the conflict is highly dependent upon the consistency of the Bayes net model as well as appropriate selection of reward function parameters. Although decomposition significantly reduces computational overhead, some drawbacks of the decomposition approach were also revealed in simulation results in the form of aggressive changes in thresholds. Otherwise, the performance of the ADP-based approach showed close comparison with the main MDP in terms of total obtained expected reward along the trajectories under the different fault cases.

Chapter 6

Mission-Based Fault Reconfiguration Framework

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 function 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 of a 1DOF spacecraft with a reaction wheel system that is on a mission to collect scientific data from three targets, as a baseline test case. We also show that there is a design tradeoff between safe operations versus mission completion. Simulation results are presented to indicate this tradeoff in the selection of design parameters for the proposed framework.

6.1 Motivation

Today's space missions are increasingly sophisticated in part due to improvements in onboard sensing and computing capabilities. One of the critical challenges for autonomous or semi-autonomous space missions is 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. [87] 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 where weight on each control law depends upon the probability of that failure as predicted by the detection scheme. Also, Abu Bakar and Veres [4] 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 chapter 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) 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 [89] 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 state that contains information about fault decisions, probability of correctness of the fault decisions, status of the mission and 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 [89] for the fault probabilities of the electronics and the reaction 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 evidence 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 a finite state-space for the MDP in which state contains information about probabilities of correctness of fault decisions.

Below, we present the problem statement, our assumptions, and our MDP formulation for fault reconfiguration. In Section 6.3 we present an implementation example of a 1DOF spacecraft attitude control system with a reaction wheel and associated electronics boards. In Section 6.4 we present simulation results and discuss the tradeoff between safe operations versus emphasis upon mission completion. Finally, we present conclusions and future work related to fault reconfiguration in Section 6.5.

6.2 Problem Formulation and Solution Approach

6.2.1 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 a joint failure probability distribution model, i.e. a Bayes net.

A7. Reconfiguration actions are executed instantaneously. This allows the assumption that ault 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.

6.2.2 MDP Formulation

We present two possible formulations for the states of the fault reconfiguration MDP. One of the formulations includes 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 a Bayes net as will be further described in the context of the spacecraft case study presented in Section 6.3.

The first formulation incorporates probabilities of fault flag correctness into the MDP state as given by:

$$
\begin{aligned}
&S = \{s_1, s_2, s_3, ..., s_N\} \\
&\textit{where}, \\
&s_i = \{A_i, B_i, F_i, P_i, sw_i, c_i\}, \\
&A_i = \{a_i^1, a_i^2, ..., a_i^{n1}\}, \\
&B_i = \{b_i^1, b_i^2, ..., b_i^{n2}\}, \\
&F_i = \{f_i^1, f_i^2, ..., f_i^{m1}\}, \\
&P_i = \{p_i^1, p_i^2, ..., p_i^{m1}\}, \\
&i \in \{1, 2, ..., N\}.
\end{aligned}
\tag{6.2.1}
$$

Here, B_i is a vector of mission objective flags representing whether or not a certain objective has been achieved. A_i is a vector of mission-related action flags indicating which of the actions are active or inactive at the moment. F is a vector of resolved fault flags generated by the fault detection process described in Chapter 5. P is a vector of probabilities of correctness of the fault flags in F. Note that there is 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 switch configuration and c is a discrete variable representing the currently active control law for the system. Note that based on assumption A6 probabilities of correctness of the fault detection are known. Further implication of A6 is that there are finitely many values for the elements of P as explained above.

The second state formulation replaces detection correctness probability vector P with abstracted sensor-based observation vector O and has the following form,

$$
\begin{aligned}
&S = \{s_1, s_2, s_3, ..., s_N\} \\
&\textit{where}, \\
&s_i = \{A_i, B_i, F_i, O_i, sw_i, c_i\}, \\
&A_i = \{a_i^1, a_i^2, ..., a_i^{n1}\}, \\
&B_i = \{b_i^1, b_i^2, ..., b_i^{n2}\}, \\
&F_i = \{f_i^1, f_i^2, ..., f_i^{m1}\}, \\
&O_i = \{o_i^1, o_i^2, ..., o_i^{m2}\}, \\
&i \in \{1, 2, ..., N\}.
\end{aligned}
\tag{6.2.2}
$$

The actions in this formulation are represented as

$$
M = \{sw_1, sw_2, ..., sw_{m3}, c_1, c_2, ..., c_{m4}, NOOP\}, \tag{6.2.3}
$$

where sw_i represents the selection of a particular switching configuration (i) among $m3$ possible switching 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, designated by *NOOP* in (6.2.3).

The reward function is defined as a negative exponential of the corresponding cost function and is given by

$$
R(s_i) = \lambda \exp\left\{ \begin{array}{l} -\sum_{k=1}^{n1} \alpha_k I\left(s_i \in A_k^{critical}\right) a_i^k - \sum_{k=1}^{n2} \beta_k I\left(s_i \in B_k^{critical}\right)\left(1 - b_i^k\right) \\ -\sum_{k=1}^{m1} G\left(f_i^k, p_i^k, sw_i, c_i\right) \end{array} \right\} \tag{6.2.4}
$$

where α_k, β_k, and λ are positive constants for all k, I is an indicator function ($I(x) = 1$ when x is true and 0 otherwise), $A_k^{critical}$ is a subset of the state space for which activation of a certain mission related action (a^k) is critical or undesired, $B_k^{critical}$ is a subset of the state

space containing those states that can lead to an inability to achieve 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 chosen so that it is zero if for the current fault flags and current probabilities of correct detection, the switching and control configuration is optimal in some sense, e.g. enables safe operation of the spacecraft,. G is positive otherwise with highest value at the worst possible configuration, e.g., the least safe operational state. Note that the reward function is an exponential of the cost function which has three main terms. The first term penalizes all the states where the active mission-related action is undesirable, where undesirable states are states to avoid when a particular mission related action is active. The second term penalizes being in states which could be undesirable for all or some of the unachieved objectives. The third and final term penalizes being in states where given the current fault flags and their probabilities of correctness the current configuration is suboptimal. The reward function is chosen as an exponential so that the reward takes non negative values.

The reward function for states with sensor observations (O) instead of pre-computed probabilities for the values of the fault flags being correct (P) can be represented as,

$$R(s_i) = \lambda \exp\left\{ \begin{array}{l} -\sum_{k=1}^{n1} \alpha_k I\left(s_i \in A_k^{critical}\right) a_i^k - \sum_{k=1}^{n2} \beta_k I\left(s_i \in B_k^{critical}\right)\left(1 - b_i^k\right) \\ -\sum_{k=1}^{m1} \overline{G}\left(f_i^k, O_i^k, sw_i, c_i\right) \end{array} \right\} \tag{6.2.5}$$

where O_i^k is the subset of O_i that is used to calculate p_i^k.

To specify transition probabilities, there are two possible cases for each of our alternate state formulations. In the first case, we assume that the actions are instantaneous with

respect to the changing fault flags and other conditions (Assumption A7). Therefore, each action could simply result either in the desired state (i.e. state with the desired configuration with everything else being the same) or the same state from which it is executed depending upon the probability of failure of the reconfiguration action. Hence the transition probability function can be represented as

$$T(s_j \mid s_i, \mu_k) = \begin{cases} \theta_1 & if : \left(\{s_j \setminus sw_j\} = \{s_i \setminus sw_i\}\right) \wedge sw_j = \mu_k \\ \theta_2 & if : \left(\{s_j \setminus c_j\} = \{s_i \setminus c_i\}\right) \wedge c_j = \mu_k \\ 1-\theta_1 & if : s_j = s_i \wedge sw_j = \mu_k \\ 1-\theta_2 & if : s_j = s_i \wedge c_j = \mu_k \end{cases} \quad (6.2.6)$$
$$\mu_k \in M$$

where θ_1 and θ_2 are probabilities of success over the switching and control law reconfiguration actions respectively. Also, $x \setminus y$ for $y \in x$ implies elements of set x not also in y.

In the second case, we allow other transitions to happen during the execution of a reconfiguration action i.e. we discard assumption A7. The resulting transition probability function is presented in Equation (6.2.7). Although the probabilities of each transition type are shown separately, any combination of the transitions can happen at a particular time. For example, given a reconfiguration action in progress, the reconfiguration may fail, but on the other hand, one of the unachieved mission objectives may still be achieved, or the sensor output could change hence changing the fault flag correctness probabilities. Alternatively the fault detection scheme may decide to change fault flags based on new sensor information, etc. Note that values of the fault flags and the probability of reconfiguration action failure depend upon sensor readings but sensor readings are considered independent of the values of fault flags. Also, the probabilities of

failures are conditionally independent of the values of the fault flags given the sensor readings.

$$T(s_j \mid s_i, \mu_k) = \begin{cases} \theta_1 & Transition(sw_i) \\ \theta_2 & Transition(c_i) \\ \sum_{l=1}^{n1} \varphi 1_l & Transition(A_i) \\ \sum_{l=1}^{n2} \varphi 2_l & Transition(B_i) \\ \sum_{l=(1,1)}^{(m1,m1)} \varphi 3_l & Transition(P_i, F_i) \\ 1-\theta_1-\sum_{l=1}^{n1}\varphi 1_l-\sum_{l=1}^{n2}\varphi 2_l-\sum_{l=1}^{m1}\varphi 3_l & NoTransition(\mu_k = sw_j) \\ 1-\theta_2-\sum_{l=1}^{n1}\varphi 1_l-\sum_{l=1}^{n2}\varphi 2_l-\sum_{l=1}^{m1}\varphi 3_l & NoTransition(\mu_k = c_j) \end{cases} \tag{6.2.7}$$

The transition function for the state formulation with O instead of P can be written similarly. Below, we assume the case of instantaneous reconfiguration actions is true (that Assumption A7 holds) and also we will assume that the states are represented with vector O instead of P. Representation with P is similar. We do not present an example for the model with transition probabilities as in Equation (6.2.7) because it requires further assumptions about the knowledge of corresponding transition probabilities e.g. φ_1, φ_2, φ_3 etc which is beyond the scope of this chapter. Nevertheless, this could be a desirable future directionof study.

6.3 Baseline Spacecraft Case Study

Consider a 1DOF spacecraft with reaction wheel where the reaction wheel has two associated electronics boards, and each electronics board has current and voltage sensors to detect failure of the electronics board. There is also an inertial measurement unit

(IMU) that can be used to detect faulty operation of the reaction wheel based on rotation rate of the spacecraft. We assume that the spacecraft is on a mission to collect scientific data from three targets. At each point, we have the information of the collected data so far. The mission-related actions are attitude maneuvering and data collection. For simplicity, we assume that the data collection equipment does not fail. Figure 6.1 shows the system under consideration.

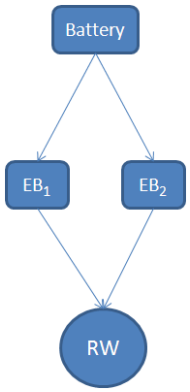

Figure 6.1: 1-DOF reaction wheel (RW) system

Note that we can generate a Bayes net for this example using the interconnection of components and information about how they work together. Figure 6.2 shows the Bayes net for this example. In Figure 6.2, nodes *Batt*, *EB*1, *EB*2, and *RW* represent faulty/normal mode of these components. Nodes O_{EB1} and O_{EB2} represent abstracted sensor readings in terms of faulty/normal for *EB*1 and *EB*2, respectively. Node SW_{EB} has values {1, 2} indicating which of the boards is in use. Node *C* represents the active control law. We assume three control laws one of which (c_1) works best when the wheel is normal, the second law (c_2) works best when the wheel is faulty, and the third law (c_3) is the safe mode control where the reaction wheel is turned off. We also assume that the control law appropriate for a faulty wheel (c_2) yields lower performance than c_1 if the

wheel is operating normally. Node a_{att} indicates whether or not the attitude maneuvering action is active. Node *IMU* represents the normal/faulty status of the wheel abstracted from *IMU* measurements. The initial conditional distributions corresponding to the Bayes Net are shown in

Table 6-1. The symbol "-" before a variable in

Table 6-1 indicates a *faulty* mode except for *SW*1 and *SW*2 where it indicates that the corresponding switch is turned *off*. Note that all probabilities in

Table 6-1 are probabilities of components being in *normal* mode given evidence from parent nodes. The corresponding probabilities of components being in *faulty* modes are calculated by subtracting probability of normal operation from 1. Using these distributions, the probabilities of failure (given any evidence) for any of the components can be calculated using Bayesian inference. We assume that the probability of failure of *RW* is 0 when no attitude maneuvering action is under execution. Also we assume that the failure/normal mode of RW does not change when control law c_2 or c_3 is in effect.

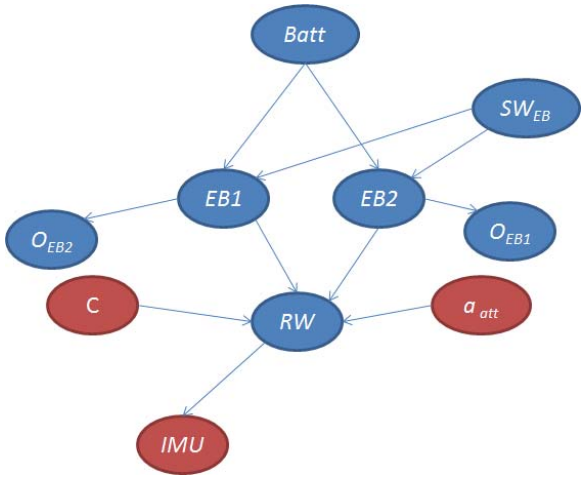

Figure 6.2: Bayes net for 1-DOF reaction wheel system

Table 6-1: Conditional probabilities for the Bayes net

P(Batt)	0.995
P(SW1)	0.5
P(SW2)	0.5
P(EB1|B,SW1)	0.998
P(EB1|-SW1,B)	0.999
P(EB1|SW1,-B)	0.4
P(EB1|-B,-SW1)	0.7
P(EB2|B,SW2)	0.998
P(EB2|-SW2,B)	0.999
P(EB2|SW2,-B)	0.4
P(EB2|-B,-SW2)	0.7
$P(O_{EB1}\|EB1)$	0.95
$P(O_{EB1}\|\text{-}EB1)$	0.15
$P(O_{EB2}\|EB2)$	0.95
$P(O_{EB2}\|\text{-}EB2)$	0.15

$P(RW\|EB1,EB2,c1,a_{att})$	0.999
$P(RW\|EB1,\text{-}EB2,c1,a_{att})$	0.999
$P(RW\|\text{-}EB1,EB2,c1,a_{att})$	0.999
$P(RW\|\text{-}EB1,\text{-}EB2,c1,a_{att})$	0.2
$P(IMU\|RW)$	0.99
$P(IMU\|\text{-}RW)$	0.02

Now we present the MDP formulation for this example. The states of the system are represented as

$$S = \{s_1, s_2, s_3, ..., s_N\}$$
$$s_i = \left\{a_i^{att}, b_i^1, b_i^2, b_i^3, f_i^{Batt}, f_i^{RW}, f_i^{EB1}, f_i^{EB2}, o_i^{EB1}, o_i^{EB2}, o_i^{IMU}, sw_i, c_i\right\} \quad (6.3.1)$$

Given the information in a baseline state, we can solve the Bayes net in Figure 6.2 to obtain probabilities of failures for *RW*, *EB*1, *EB*2, and *Batt*. Then we can compare probabilities of failure with values of fault flags in the MDP states to determine probabilities of false alarms and missed detections. The total number of states in above equation amounts to 12,228. Since we have 6 actions, the computational complexity of this example will be of the order of 10^8 which is manageable with a modern computer.

Available actions are to flip a switch and to flip the control law. Doing nothing (NOOP) is also among available options in action set *M* given by

$$M = \{sw_1, sw_2, c_1, c_2, c_3, NOOP\} \quad (6.3.2)$$

The reward function requires specification of critical sets and a *G* function that represents the penalty on incorrect reconfiguration. We assume the attitude maneuvering

action is not desirable in states where the control law is c_2 or c_3 and f^{RW} is 0 (indicating no fault in the *RW*). Furthermore, an attitude maneuver can be executed with either c_1 or c_2 as the control law whereas, with c_3, attitude maneuvering is not possible. Also we assume that states with c_2 or c_3 as the active control law are critical for achieving objective b^3 (i.e. collecting data from asteroid 3). This may be due to pointing stability requirements for this objective. For the other two objectives, the critical states are when the control law is equal to c_3. We define the *G* function for our example as

$$G(s_i)=\left\{\begin{array}{l}\lambda_1\left[P(MD)_i^{EB1}\left(1-f_i^{EB1}\right)+\left(1-P(FA)_i^{EB1}\right)f_i^{EB1}\right]I\left(sw=1\right)\\+\lambda_2\left[P(MD)_i^{EB2}\left(1-f_i^{EB2}\right)+\left(1-P(FA)_i^{EB2}\right)f_i^{EB2}\right]I\left(sw=2\right)\\+\lambda_3\left[\left(1-P(MD)_i^{RW}\right)\left(1-f_i^{RW}\right)+P(FA)_i^{RW}\,f_i^{RW}\right]I\left(c=2\right)\\+\lambda_4\left[\left(1-P(MD)_i^{RW}\right)\left(1-f_i^{RW}\right)+P(FA)_i^{RW}\,f_i^{RW}\right]I\left(c=3\right)\\+\lambda_5\left[P(MD)_i^{RW}\left(1-f_i^{RW}\right)+\left(1-P(FA)_i^{RW}\right)f_i^{RW}\right]I\left(c=1\right)\\+\lambda_6\left[P(MD)_i^{Batt}\left(1-f_i^{Batt}\right)+\left(1-P(FA)_i^{Batt}\right)f_i^{Batt}\right]\left(1-I\left(c=3\right)\right)\end{array}\right\} \tag{6.3.3}$$

In the above equation, $P(FA^j{}_i)$ and $P(MD^j{}_i)$ represent probabilities of false alarm and missed detection respectively for component *j* in state *i*. These probabilities are conditioned upon the information given in state s_i and are calculated from the Bayes net in Figure 6.2 as described earlier. Also λ_1, λ_2, λ_3, λ_4, λ_5, and λ_6 are positive weighting factors. Note that, function *G* has six main terms. The first two terms in *G* penalize usage of a faulty electronics board. λ_1 and λ_2 are different because the two boards in general may not be exact copies of each other. The next two terms in *G* penalize usage of degraded control laws when the reaction wheel is still healthy. The fifth term penalizes the usage of the normal control law c_1 when the wheel is faulty. The sixth term in *G* penalizes not using safe mode control in case of battery failure.

Now, we can write the reward function for our example as

$$R(s_i) = \lambda \exp\left\{ \begin{array}{l} -\alpha_1 I\left(s_i \in A_{att}^{critical}\right) a_i^{att} - \sum_{k=1}^{3} \beta_k I\left(s_i \in B_k^{critical}\right)\left(1 - b_i^k\right) \\ -G(s_i) \end{array} \right\} \tag{6.3.4}$$

Finally, transition probabilities for the example are computed using Equation (6.2.6) with $\theta_1 = 0.9$ and $\theta_2 = 1$.

Note that there are no constraints in the specified 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 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 (6.2.7) 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.

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

6.4 Simulation Results

In this section we present simulation results that emphasize the importance of selecting design parameters and the tradeoff between safe operations versus mission completion. We present two case studies. In the first case, parameters are selected to emphasize mission completion and in the second case, parameter selection favors safe operation.

6.4.1 Case 1: Emphasizing Mission Completion

For this case, we selected $\alpha = 2$, β_1, β_2, and β_3 equal to 9, $\lambda = 10^5$, λ_1, λ_2, and λ_3 equal to 3, λ_4 and λ_6 equal to 4, and $\lambda_5 = 5$. Also we selected discount factor $\gamma = 0.8$. In our simulation we started with an initial state where the spacecraft was in safe mode. Table 6-2 shows the response of the MDP policy to various events. Note that each state in Table 6-2 has the format $s_i = \left\{a_i^{att}, b_i^1, b_i^2, b_i^3, f_i^{Batt}, f_i^{RW}, f_i^{EB1}, f_i^{EB2}, o_i^{EB1}, o_i^{EB2}, o_i^{IMU}, sw_i, c_i\right\}$ where the values of fault flags are 1 when the component is faulty whereas the values of abstracted observation flags are 1 when the component is observed as healthy.

Table 6-2: State trajectory emphasizing mission completion.

Simulation Step #	State	Policy	Exogenous event
1	[0 0 0 0 0 0 0 0 1 1 1 1 3]	c_1	attitude maneuver begins
2	[1 0 0 0 0 0 0 0 1 1 1 1 1]	NOOP	O_{EB1} indicates failure
3	[1 0 0 0 0 0 1 0 0 1 1 1 1]	sw_2	attitude maneuver ends
4	[0 0 0 0 0 0 1 0 0 1 1 2 1]	NOOP	mission objective 1 completes
5	[0 1 0 0 0 0 1 0 0 1 1 2 1]	NOOP	attitude maneuver begins
6	[1 1 0 0 0 0 1 0 0 1 1 2 1]	NOOP	O_{EB2} indicates failure
7	[1 1 0 0 0 0 1 0 0 0 1 2 1]	NOOP	*IMU* indicates failure
8	[1 1 0 0 0 0 1 0 0 0 0 2 1]	NOOP	attitude maneuver ends
9	[0 1 0 0 0 0 1 0 0 0 0 2 1]	NOOP	mission objective 3 completes
10	[0 1 0 1 0 0 1 0 0 0 0 2 1]	c_2	attitude maneuver begins
11	[1 1 0 1 0 0 1 0 0 0 0 2 1]	NOOP	attitude maneuver ends
12	[0 1 0 1 0 0 1 0 0 0 0 2 1]	NOOP	mission objective 2 completes
13	[0 1 1 1 0 0 1 0 0 0 0 2 1]	c_3	---

Several observations are worthy of note. First, in step 7, although both electronics boards have been diagnosed as failed, the policy insists on keeping control law c_1 to avoid entering critical states for any of the remaining objectives. In step 10, the control law is changed to c_2. This is because c_2 is the safest controller among the ones that are feasible for the remaining mission objective. In step 11, the policy switches to the state that is

infeasible for the attitude maneuver since f^{RW} is 0 and c_2 is activated; this is because a low value of α is selected as compared to safety-related parameters. After all the objectives are completed, the reconfiguration policy opts for the safest control law i.e. c_3. This example raises the question of spacecraft safety. What if the calls made by O_{EB2} and *IMU* were correct? In that case, the spacecraft would not have completed its objectives and the control law would have stayed at c_1 forever. This fact points towards a need for an external safing mechanism to avoid consequences of a mission emphasizing policymission completion to the extent that the mission can actually be jeopardized (unsafe). On the other hand, the response of the policy is justified because the user has defined his preferences through design parameters which indicate that mission objectives are more important than safety.

6.4.2 Case 2: Emphasizing safety

For this case, we selected $\alpha = 2$, β_1, β_2, and β_3 equal to 3, $\lambda = 10^5$, λ_1, λ_2, and λ_3 equal to 3, λ_4 and λ_6 equal to 4, and $\lambda_5 = 5$. Also we selected discount factor $\gamma = 0.8$. We started with an initial state where the spacecraft was in safe mode. Table 6-3 shows the response of the MDP policy to various events.

Table 6-3: State trajectory with safety emphasizing policy

Simulation Step #	State	Policy	Exogenous event
1	[0 0 0 0 0 0 0 0 1 1 1 1 3]	c_1	attitude maneuver begins
2	[1 0 0 0 0 0 0 0 1 1 1 1 1]	NOOP	O_{EB1} indicates failure
3	[1 0 0 0 0 0 1 0 0 1 1 1 1]	sw_2	attitude maneuver ends
4	[0 0 0 0 0 0 1 0 0 1 1 2 1]	NOOP	mission objective 1 completes
5	[0 1 0 0 0 0 1 0 0 1 1 2 1]	NOOP	attitude maneuver begins
6	[1 1 0 0 0 0 1 0 0 1 1 2 1]	NOOP	O_{EB2} indicates failure
7	[1 1 0 0 0 0 1 0 0 0 1 2 1]	NOOP	*IMU* indicates failure
8	[1 1 0 0 0 0 1 0 0 0 0 2 1]	c_3	attitude maneuver ends (unsuccessful)

Note that, as opposed to the Table 6-2 trajectory, the trajectory in Table 6-3 goes into the safe controller even though the mission objectives have not yet been achieved. At step 7, although there was an indication both electronics boards have failed, the policy waited for the signal from the *IMU* and when sufficient evidence of failure was received, the spacecraft control law was changed to a safe one.

From the above two cases, there is a clear tradeoff between emphases upon mission completion versus safe operation. It is important to point out here that having high values for parameters related to both safety and mission-critical states is not the answer to this problem because what really matters is their relative weighting given that tradeoffs are ultimately required.

6.5 Complexity Analysis and ADP

An ADP method similar to the methods presented in chapters 4 and 5 can be used to reduce the computational complexity of the problem, although we reserve specifics of ADP application to spacecraft fault reconfiguration for future work. Usually, there are cases where certain components (or subsystems) can be reconfigured independently of the other components (or subsystems). There may also be hierarchical relationships in reconfiguration that can be used to separate lower level reconfiguration from reconfiguration at higher layers of abstraction. Also, as in our case study in this chapter, the reconfiguration problem may often be static which allows the use of greedy search which is computationally less expensive than value iteration.

6.6 Conclusions and Future Work

We have presented a framework for calculating an optimal policy for mission-based post-fault reconfiguration. Our framework is robust in a sense that while calculating an 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 also shown a way to implement our framework and have indicated some important mission completion versus safe operation tradeoffs through 1DOF case study and simulation results. The primary drawback of our framework is its potential for high computational complexity. In the future, we would like to develop approximate dynamic programming techniques for this framework to reduce the computational overhead. For example, with ADP applied to the case of a 3-DOF spacecraft with three reaction wheels, the electronics board switching reconfiguration policy can be implemented for each wheel separately whereas the control law reconfiguration policy can be implemented at a higher level when the switching is not feasible or applicable.

Chapter 7

Far Ultraviolet Spectroscopic Explorer Case Study

In this chapter, we present a case study that is inspired by the Far Ultraviolet Spectroscopic Explorer (FUSE) mission. This mission consisted of a low earth orbit spacecraft that suffered from multiple failures related to the attitude control system. As a result useful mission time was wasted while engineers and scientists on the ground station were determining appropriate reconfiguration strategies to handle the failures. Our purpose for presenting this case study is to elaborate how technologies such as CFT-SOAP and alternative architectures ASPEN and Livingstone summarized previously in Chapter 2 could have modeled and managed reconfiguration options for attitude control such that mission downtime would have been minimized through automatic response to the encountered spacecraft hardware failures. This of course would have required anticipation of these failures, including onboard software to detect hardware problems as well as alternate control laws capable of adequately controlling FUSE spacecraft attitude when encountering any subset of the anticipated failures. CFT-SOAP application to FUSE fault management is followed by an example application of ASPEN and Livingstone for this purpose, enabling comparison of the three architectures for a real-world spacecraft application. To our knowledge attitude fault tolerance has not been modeled previously for any of the three architectures.

7.1 FUSE Mission Review

The FUSE satellite was launched in June 1995 [65] in a low earth near-circular orbit (eccentricity = 0.001) with altitude of approximately 762 kilometers and an inclination of 25 degrees. The primary objective of this satellite was to observe light in the far-ultraviolet spectral region, 905-1187 Å, with a high spectral resolution. Science instrumentation consisted of four co-aligned prime-focus telescopes and Rowland spectrographs with micro-channel plate detectors. Two of the telescope channels used Al : LiF coatings for optimum reflectivity between approximately 1000 and 1187 Å, and the other two channels used SiC coatings for optimized throughput between 905 and 1105 Å. Details of the design and early performance of FUSE can be found in [91], [90].

The FUSE Attitude Control System (ACS) consisted of two sets of three ring-laser gyroscopes (Inertial Reference Units, or IRUs) for attitude estimate propagation. Redundant three-axis magnetometers and coarse sun sensors provided coarse attitude information to ±2°. The required attitude resolution for fine pointing was achieved by using a signal from a Fine Error Sensor [59] (FES) in the science instrument which images a region of the sky around the spectrograph apertures. Four Reaction Wheel Assemblies (RWAs) were used to control the attitude of the satellite and manage angular momentum. Three wheels were arranged along the primary roll, pitch, and yaw axes of the satellite and a fourth skew wheel was oriented equidistant from the others. The skew wheel was biased to minimize zero-speed crossings on the other wheels and could serve as a substitute in case of failure of one of the other RWAs. Three magnetic torqueing bars (MTBs) were mounted along the primary axes and were used to control the momentum load on the wheels by inducing torque on the spacecraft from the Earth's magnetic field.

A more detailed description of the design of the ACS, along with a description of its performance early in the mission and in two-wheel mode is available in [56], [65].

Beginning in late 2001, the failure of spacecraft components began to affect satellite operations. While two out of the four reaction wheels failed by the end of 2001, the flight software was modified to employ the torqueing bars in conjunction with the two remaining wheels to provide fine pointing control. This upgrade in the software required engineers and scientists to work round the clock for about 54 days. At this point, additional software was also being developed for the cases where gyroscopes might fail. In December 2004, the third reaction wheel also failed rendering the spacecraft into the safe mode once again and requiring major changes in the FUSE mission planning and attitude control. Regular scientific operations resumed in November 2005. By mid-2006, the satellite was operating with only one of its four reaction wheels and two of its six gyroscopes [91]. Loss of these hardware components required a significant redesign of the ACS, but with software revisions manually computed by engineers on the ground, FUSE was restored to operation, making observations with an efficiency approaching that of its earlier days, albeit over a smaller portion of the sky.

7.2 Reliability Prediction (The Probabilities of Failures)

Application of CFT-SOAP for fault-tolerant attitude control in the FUSE mission requires specification of a priori failure probabilities for relevant spacecraft components and systems. These probabilities can be estimated from available failure databases compiled over previous spacecraft missions, as well as through component-level testing by the manufacturers. Failures observed during spacecraft missions have been compiled in [96], [20], and [19]. In [96], a survey on serviceable spacecraft failures is presented

with a failure database including 854 failure records spanning the years 1957 to 2000. This paper also presents a list of 242 partial and total failures out of 2431 space missions between years 1981 and 2000. In [47] 3000 anomalous incidents are analyzed to predict spacecraft reliability. In 80% of these incidents further analysis was possible to determine the cause of the failure. Data was obtained from over 300 satellites launched between the early 1960s through January of 1984. In [20] the authors have presented a reliability analysis for 1584 earth-orbiting satellites launched between January 1990 and October 2008. Since its data is most recent and is obtained from Earth-orbiting spacecraft missions, the data presented in [20] is used in this chapter to estimate the probability of failures for gyroscopes and reaction wheels in the FUSE spacecraft. According to [20] the reliability of gyroscopes and reaction wheels is approximately 99.5% after 4 years on-orbit. Therefore, prior probability of failure within the first 4 years is 0.005 [100]. The contribution of each sub-system to spacecraft failure in terms of pie-charts for different durations spent on orbit is presented in [19]. According to these pie charts, gyroscope and reaction wheel failures contribute about 11% to the total failures after 5 years of launch. This is consistent with [47] where the percentage of failures due to guidance and navigation is determined to be approximately 12% for science missions. Also, in [96], the total number of guidance and navigation failures is 10% of the total number of cases considered. If we assume that 10% of these failures are due to gyroscope and reaction wheels then we can conclude that the reliability of gyroscopes and reaction wheels is 99% which is reasonably close to 99.5% as depicted in [19]. Note that while FUSE components had a much higher probability of failing, mission engineers did not know these components would have a high failure rate before launch, so failure probability

models embedded in a system such as CFT-SOAP would have initially been derived from historical data as is described in this chapter.

7.3 Fuse Modeling with CFT-SOAP

To model the FUSE satellite mission using CFT-SOAP, we assume there is a multiple-model fault detection framework such as that described in [112] (also see section 5.4) onboard the FUSE spacecraft. This fault detection module must be able to detect failures in any of the reaction wheels as a minimum. The accuracy of this fault detection framework depends upon the selection of appropriate detection thresholds. The probabilities of false alarms and missed detections are assumed to be known functions of these thresholds. We also assume that there are temperature and current/voltage sensors in the electronics associated with the reaction wheels and the gyroscopes that can provide health status data for these components. We also assume that there are 15 control laws on-board the FUSE spacecraft, including the nominal (no-failure) control law, four control laws using combinations of three out of the available four reaction wheels, six control laws using combinations of two reaction wheels and magnetic torque bars, and four control laws relying on only one of the four reaction wheels and the magnetic torque bars. This is the minimal set of attitude control laws required to cover the suite of zero to three reaction wheel failure cases.

Under the above assumptions, the state space for the FUSE CFT-SOAP MDP is defined follows:

$$S = \{s_1, s_2, ..., s_N\}$$

where

$$s_i = \{A_i, B_i, v_i, z_i, Bl_i, Bo_i, O_i, V_i, sw_i, c_i, SE_i, EL_i\}$$

$$A_i \in \{0,1,2\}, B_i = \{b_i^1, b_i^2, ..., b_i^5\}, b_i^j \in \{0,1\}$$

$$v_i \in \{1,11,21,...,351\}, z \in \{1,2,3,4,5\}$$

$$Bl_i = \{bl_i^1, ..., bl_i^{10}\}, bl_i^j \in \{0,1\}, Bo_i = \{bo_i^1, ..., bo_i^4\}, bo_i^j \in \{0,1\}$$

$$O_i = \{o_i^1, ..., o_i^{10}\}, o_i^j \in \{0,1\}, V_i = \{v_i^1, ..., v_i^{14}\}, v_i^j \in \{1,2,3\}$$

$$sw_i = \{sw_i^{roll}, sw_i^{pitch}, sw_i^{yaw}\}, sw_i^j \in \{1,2\}, c_i \in \{1,2,...,14\}, SE \in \{1,2,3\}, EL \in \{1,2,...,5\}$$

(7.3.1)

In the above equation, variable *A* represents the status of in-progress mission activities where *A* = 1 if an attitude maneuver is in progress, *A* = 2 if data collection from a target region is in progress, and *A* = 0 if no mission-related action is in progress. Vector variable *B* represents the status of the mission objectives where each component is a binary flag corresponding to a particular mission objective. For FUSE, mission objectives are defined as far ultraviolet radiation observations obtained from particular celestial objects or regions. We model five specific observing targets or objectives in this case study.

In Equation 7.3.1, variable *v* represents spacecraft true anomaly assuming the remaining orbital elements are constant. We have discretized true anomaly into 10 degree regions yielding a 36-value set. The variable *z* represents attitude pointing of the spacecraft in terms of the five target regions of the sky. Vector *Bl* contains logic-based fault flags for the six gyroscopes and four reaction wheels that are generated using available information about the observed (using current/voltage sensors) operational mode of these components in vector *O* and fault information generated using dynamics-based fault flags for the four reaction wheels in vector *Bo*. Vector variable *V* contains

thresholds for fault models in the multiple-model dynamics-based fault detector. Each threshold corresponds to one of the 14 possible failures configurations for the four reaction wheels. In our case study, each threshold can have three possible values *low*, *medium*, and *high* where *high* corresponds to $v_i^j = 3$. These thresholds affect the values of fault flags in *Bo* where each flag may be affected by multiple thresholds depending upon number of fault scenarios in which that fault flag is involved e.g. fault flag for reaction wheel 1 is involved in all (one-wheel, two-wheel, three-wheel, and four-wheel) fault scenarios where failure of reaction wheel 1 is included. Vector *sw* represents which of the gyroscopes are in use along each axis of rotation. The three binary flags in *sw* represent the active gyroscope along the relevant axis (e.g. roll axis) from one (e.g. $sw^{roll} = 1$) or the other (e.g. $sw^{roll} = 2$) set of onboard gyroscopes. The active control law is represented by *c* where each value of *c* corresponds to one of the 15 possible combinations of the four reaction wheels chosen at the most three and at least one at a time. Variable *SE* represents a combination of eclipse and sun-pointing flags. *SE* = 1 means that the spacecraft is in eclipse, *SE* = 2 means that the spacecraft is not in eclipse but the telescopes are pointed towards the sun, and *SE* = 3 means that the spacecraft is not in eclipse and the telescopes are not pointed toward the sun. This variable is used to maintain safety of the instrumentation which might be damaged if open (in use) and pointed toward the sun. We plugged in the parameters of the FUSE orbit in the Satellite Tool Kit (STK) software and calculated that FUSE remained in eclipse during 33% of its orbital rotation. Later in our simulation trajectory, we assume that the eclipse starts at 230 degrees of true anomaly and ends at 350 degrees of true anomaly. Note that our CFT-SOAP formulation can deal with any location of the eclipse. Finally, variable *EL* represents energy in terms of the

charge level available in the spacecraft batteries over 5 levels where $EL = 5$ means fully charged and $EL = 1$ means lowest acceptable charge level.

With the above definition of the state-space, the total number of MDP states is approximately 1.5×10^{23}. This state-space size is too large to be considered by an integrated MDP. Therefore, we decompose the CFT-SOAP MDP into smaller MDPs dedicated to mission planning, fault detection, and reconfigurable control in the manner described in previous chapters e.g. Sections 3.4, 4.5, 5.10, and 5.11. Figure 7.1 shows the process of decomposing with CFT-SOAP for the FUSE case study. We start with the integrated approach and then decopose the larger MDPs into smaller MDPs step by step. This way, it is easier to understand the tradeoff between computational complexity and optimality of the solution since the integrated approach incorporates all the dependencies between the planning, fault detection, and reconfiguration subproblems. Figure 7.2 shows the specific map of MDPs generated by decomposing the larger MDPs in planning, fault detection and reconfiguration. The details of decomposition and the corresponding effects on the solution are discussed in the subsequent sections.

Remark: Note that the symbol for true anomaly (ν) and the components of the threshold vector V look the same but these can be differentiated from each other since components of the threshold vector V have additional index representing their identification within V.

Figure 7.1: The process of decomposition using the CFT-SOAP framework

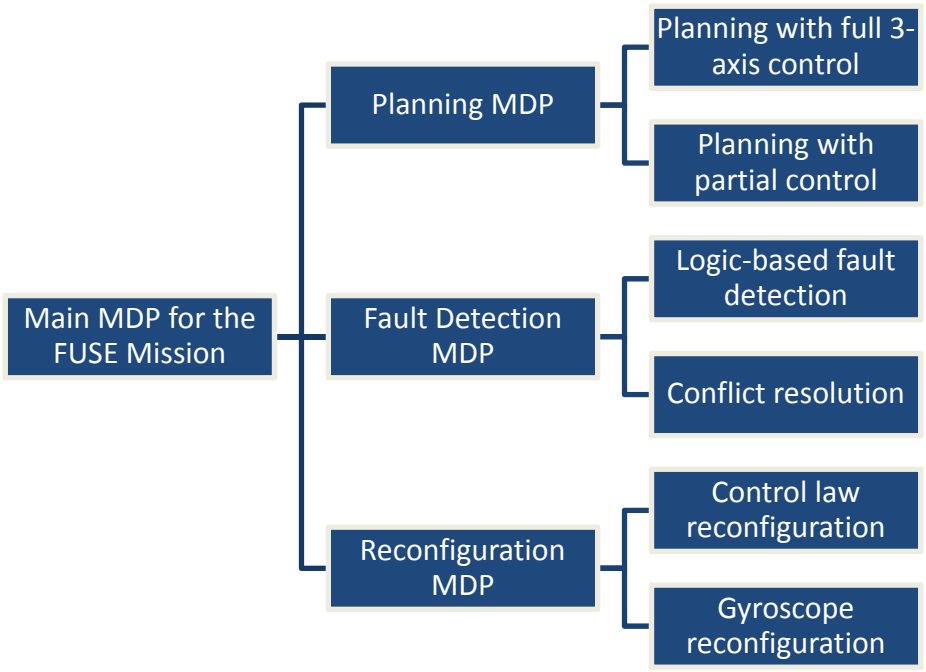

Figure 7.2: MDP decomposition map for the FUSE case study

7.3.1 The Planning MDPs

The planning MDP is tasked with sequencing observations in a manner that is feasible given the detected fault state and reconfigurable control strategy determined by the complementary fault detection and reconfiguration MDPs. For planning, we must consider the available sensors/actuators and the probabilities of failures. This may require a planning MDP for each of the failure cases, yielding a total 15 MDPs considering only reaction wheels to be the important components for mission completion since gyroscopes only help in the determination of the angular velocities that can be determined alternatively from the attitude measurements provided by the three-axis magnetometers, the course sun sensors, and the fine error sensor. We simplify our case by considering available control authority instead of available components. This simplification is done mainly to contain the size of this chapter and to demonstrate the concept of CFT-SOAP

implementation with minimum space occupation. With this simplification, we need two classes of planning MDPs, one class of MDPs for the cases with full control availability where at least two of the four reaction wheels are functional and the second class of MDPs for partial control availability where only one out of the four reaction wheels is functional. Note that, in this chapter, we do not consider the case where all four reaction wheels have failed and the control authority is provided by only magnetic torque bars. The only reason for excluding this case is that it adds complexity and is not required to demonstrate CFT-SOAP in the context of the FUSE case study.

In this chapter we focus on the planning MDPs with full control authority since the planning MDP with partial control authority can be developed using a similar formulation with additional information in the state space regarding which actions are possible and when they can be achieved (e.g., slewing may require that the magnetic torqueing bars pass through a specific part of the magnetosphere). Available actions given partial control availability will depend upon which of the reaction wheels remain healthy or if only the magnetic torqueing bars are available for control authority.

The states for the planning MDP with full control availability are defined as follows.

$$
\begin{aligned}
&S = \{s_1, s_2, ..., s_{N1}, s_F\} \\
&\textit{where} \\
&s_i = \{B_i, v_i, z_i, SE_i, EL_i\}, \forall i \in \{1, 2, ..., N1\} \\
&B_i = \{b_i^1, b_i^2, ..., b_i^5\}, b_i^j \in \{0,1\} \\
&v_i \in \{1, 11, 21, ..., 351\}, z \in \{1, 2, 3, 4, 5\} \\
&SE \in \{1, 2, 3\}, EL \in \{1, 2, ..., 5\}
\end{aligned}
\tag{7.3.2}
$$

Note that the size of the state space for this MDP is 86,400 which is manageable with currently-available computational resources. State s_F represents the failure state which represents the conditions under which the spacecraft no longer possesses full control availability indicating a switch to a policy applicable for cases of partial control availability is required.

The actions for the planning MDP with full control availability are

$$M = \{\mu_1, \mu_2, ..., \mu_5, NOOP_{10}, NOOP_{20}, NOOP_{50}\} \tag{7.3.3}$$

These actions are defined such that there is one action corresponding to each of the five target regions. Every action has an associated change in true anomaly that represents the time required to complete that action (Table 7-1). This change in true anomaly and the result of executing an action μ_i corresponding to the target *i* depends upon the state from which the action is executed. If the action is executed from a state where the spacecraft is not already pointing towards the target *i*, then the action results in the attitude maneuver from the current pointing towards the pointing *i*. If on the other hand, the spacecraft is already pointed towards the target *i* and the true anomaly of the spacecraft is within the window of visibility of the target (see chapter 4), then the action results in the collection of data from the target if the data flag for that target is set to 0 in the current state. In all other cases, the action simply results in the change in true anomaly associated with it. There are three actions of type *NOOP* with associated changes in true anomaly of 10, 20, and 50 degrees. These values were selected for *NOOP* of *short*, *medium* and *long* durations respectively since 10 degrees is the smallest possible change in true anomaly for the MDP formulation and 50 degrees is comparable to one of the largest maneuvers in

Table 7-1. Table 7-1 provides information about change in true anomalies associated with the actions and the energy required in terms of the number of charge levels of the batteries. The energy required for each attitude maneuver is 2 units and for each data collection action is 1 unit. The changes in true anomaly incurred by the attitude maneuvers are calculated from the magnitude of the maneuvers. The magnitudes of the maneuvers are calculated from the locations of the science targets in the sky with respect to an inertial frame of reference that has the origin in terms of roll, pitch, and yaw angles placed such that the spacecraft is pointed towards Target 1 at the origin. Target 2 is placed at positive (counterclockwise) 20 degrees roll rotation with respect to Target 1. Target 3 is placed at positive 40 degrees pitch rotation with respect to Target 1. Target 4 is placed at positive 30 degrees yaw rotation with respect to Target 1. Target 5 is at a location that is at positive 20 degrees roll rotation with respect to Target 4. The magnitude of the maneuver between any two targets is calculated from $cos^{-1}(k/2)$ where k is one less than the trace of the rotation matrix [51] involved in the rotation between the two targets. Note that the changes in true anomaly are the same no matter where in the orbit the action is applied. This is the result of the fact that we model the FUSE orbit as circular (in real-world deployment the orbit had eccentricity = 0.001).

Table 7-1: True anomaly changes and data collection windows

For current Attitude pointing	**Associated change in true anomaly (deg) for attitude maneuvering/data collection and (energy required)**					**Data collection window (deg)**
	μ_1	μ_2	μ_3	μ_4	μ_5	
1	10 (1)	20 (2)	40 (2)	30 (2)	40 (2)	50-100
2	20 (2)	10 (1)	50 (2)	40 (2)	30 (2)	100-150
3	40 (2)	50 (2)	10 (1)	50 (2)	50 (2)	150-200
4	30 (2)	40 (2)	50 (2)	10 (1)	20 (2)	200-250
5	40 (2)	30 (2)	50 (2)	20 (2)	10 (1)	250-300

The diagonal entries in columns 2 through 6 of Table 7-1 indicate changes in true anomalies associated with data collection. All other entries in these columns are associated with attitude maneuvers between the targets. The numbers in parentheses indicate the energy required for each of the attitude maneuver/data collection actions. Note that we assume here that energy is consumed by an action only if the spacecraft is in eclipse. This means that the rate of energy consumption is equal to the rate of energy production when the spacecraft is executing a mission related action while not in eclipse. Also, if no mission-related action is being performed, energy is produced at the rate of 1 unit per 10 degrees change in true anomaly. The last column of Table 7-1 represents data collection windows for targets 1 through 5 in descending order.

The reward function for the planning MDP with full control authority is defined as

$$R(s_i) = \left\{ \left(\sum_{k=1}^{n} b_i^k r_k \right) + \frac{1}{2} r_{z_i} (1 - b_i^{z_i}) + 500 \prod_{k=1}^{n} b_i^k : i \in \{1, 2, ..., N1\} \right\}$$

where

$$r = \{r_1, r_2, r_3, r_4, r_5\} = \{30, 50, 70, 40, 60\} \tag{7.3.4}$$

The costs of the actions are defined to be proportional to the associated changes in the true anomalies and have values equal to one-half of the entries in Table 7-1.

Transition probabilities are calculated from the failure probabilities of the components using [19], [20] (per Section 7.2) along with the change in true anomaly associated with each action. The failure probabilities are calculated by using the assumption of independence i.e. failure probability of each component is independent of the failure

probability of any other component. Also, for *NOOP* actions, failure probabilities are assumed to be zero. Since control authority is associated with the reaction wheels in our case study, and since at least three out of four wheels have to fail to reach a partial control availability state (state s_F in Equation 7.3.2), the probability of reaching s_F becomes 1.25×10^{-7}. We multiply this number with the associated change in true anomaly and a safety factor of 100 yielding 1.25×10^{-5} times the change in true anomaly incorporated by the given mission-related action. The safety factor is used to compensate for the fact that we have aggregated failure cases to enable separation of the planning MDP from details of fault detection and control reconfiguration.

7.3.2 The Fault Detection MDPs

In our FUSE case study, we only consider the faults in the reaction wheels and gyroscopes. Therefore, there are 10 independent fault flags, six for gyroscopes and four for the reaction wheels. In our formulation, we assume that reaction wheel faults are detectable by both dynamics-based and logic-based methods whereas ring laser gyroscope faults are detectable only by the logic-based method. This assumption is made for two reasons; first, sensor fault detection using the dynamics-based model is not straightforward and second, the laser gyroscopes are easier to diagnose by sensing the laser light intensity than by comparing the output angular velocity with a dynamic model. Therefore, the MDP state-space for fault detection is defined as follows

$$
\begin{aligned}
&S = \{s_1, s_2, ..., s_{N3}\} \\
&\textit{where} \\
&s_i = \{Bl_i, Bo_i, O_i, V_i\} \\
&Bl_i = \{bl_i^1, ..., bl_i^{10}\}, bl_i^j \in \{0,1\}, Bo_i = \{bo_i^1, ..., bo_i^{10}\}, bo_i^j \in \{0,1\} \\
&O_i = \{o_i^1, ..., o_i^{10}\}, o_i^j \in \{0,1\}, V_i = \{v_i^1, ..., v_i^{14}\}, v_i^j \in \{1,2,3\}
\end{aligned} \tag{7.3.5}
$$

Recall that vector *Bl* contains logic-based fault flags, vector *Bo* includes observer-based fault flags, vector *O* includes processed observation flags (or observed component modes), and vector *V* is the set of thresholds for the observer-based fault detector. With the above definition, the size of the state-space becomes approximately 8×10^{13}. Since this size is still formidable, we decompose the fault detection MDP into two MDPs. The first MDP is responsible for generating an optimal policy for logic-based fault detection of the reaction wheels and the gyroscopes. The second MDP is responsible for conflict resolution between logic-based and dynamics-based fault flags for the reaction wheels by controlling the thresholds for the dynamics-based fault detector. We assume the existence of an observer module capable of providing dynamics-based fault information to our conflict resolution MDP.

7.3.2.1 Logic-based Fault Detection MDP

This MDP is defined using the assumptions that all four reaction wheels have identical chances of failure and at any given time, at most three out of four wheels are in use. Similarly, we assume that all six gyroscopes have identical chances of failure and at any given time, at most 3 out of 6 gyroscopes are in use. Therefore, we need to detect the faults for only in-use reaction wheels and gyroscopes. Note that the fault flags exist for all four reaction wheels and all six gyroscopes (total 10 flags) but at any given time, only

six of these fault flags are used by the fault detection MDPs based on which of the components are in use versus not in use. This implies the existence of onboard logic to map the three reaction wheels in-use to the three flags in the MDP policy. The assumption of identical chances of failures for all four wheels and for all six gyroscopes is important for this formulation. The states for the resulting logic-based fault detection MDP are defined as

$$
\begin{aligned}
&S = \{s_1, s_2, ..., s_{N4}\} \\
&\textit{where} \\
&s_i = \{Bl_i, Bo_i, O_i\} \\
&Bl_i = \{bl_i^1, ..., bl_i^6\}, bl_i^j \in \{0,1\}, Bo_i = \{bo_i^1, ..., bo_i^3\}, bo_i^j \in \{0,1\} \\
&O_i = \{o_i^1, ..., o_i^6\}, o_i^j \in \{0,1\}
\end{aligned}
\tag{7.3.6}
$$

The size of the state space with above definition is 32,768. The actions for this MDP are defined as

$$
M = \{\mu_1, \mu_2, ..., \mu_6, NOOP\} \tag{7.3.7}
$$

where six of the actions correspond to the switching (on/off) of logic-based fault flags where the first three actions are for gyroscopes and the next three actions are for reaction wheels. The axis order for both gyroscopes and reaction wheels is yaw, pitch, and roll. In case the slew axis reaction wheel is in-use, its corresponding action is always μ_6 and the remaining wheels are assingned μ index according to the yaw, pitch, and roll precidence. The last action is for no-operation. The reward function for this MDP is similar to the one presented in Section 5.10.1 and is represented as

$$R(s_i) = \lambda e^{-J(s_i)}$$

where,

$$J(s_i) = \left\{ \begin{array}{c} q_1 \sum_{k=1}^{6} \{P(FA_k \mid Bo_i, O_i) + P(MD_k \mid Bo_i, O_i)\} \\ + \sum_{k1,k2 \in \{4,5,6\}} q_2 \left| bl_i^{k1} - bo_i^{k2} \right| \end{array} \right\} \quad (7.3.8)$$

where $q_1 = q_2 = 1$, and the probabilities of missed detections and false alarms are calculated from Table 7-2. The probabilities in the table are multiplied by bl_k and (1- bl_k) to indicate that the probability of false alarm is nonzero only when the corresponding fault is identified as present. Similarly, the probability of missed detection is nonzero only when the corresponding fault is determined to not exist.

Table 7-2: False alarm and missed detection probabilities for the logic-based fault detection

Value of bo_k	Value of o_k	For the Reaction Wheels		For the Gyroscopes	
		P(FA_k|bo_k,o_k)	P(MD_k|bo_k,o_k)	P(FA_k|bo_k,o_k)	P(MD_k|bo_k,o_k)
0	0	0.083bl_k	0.917(1- bl_k)	0.01 bl_k	0.99(1- bl_k)
0	1	0.999bl_k	0.001(1- bl_k)	0.99 bl_k	0.01(1- bl_k)
1	0	0.001bl_k	0.999(1- bl_k)	0.01 bl_k	0.99(1- bl_k)
1	1	0.917bl_k	0.083(1- bl_k)	0.99 bl_k	0.01(1- bl_k)

All actions in this MDP are deterministic therefore there is no transition probability matrix involved. Also, since all the actions in this MDP are instantaneous and available from every state, the optimal policy for this MDP can be calculated using greedy search instead of value iteration to reduce computational overhead.

7.3.2.2 Conflict Resolution MDP

For this MDP, we make the same assumptions that all four reaction wheels have identical chances of failure and that reaction wheels can fail at any time (any true anomaly). We

also assume as described above that at most three out of four reaction wheels are in operation. Also recall that the gyroscope failures are only determined by the logic-based fault detection scheme. Therefore, conflict resolution has to be performed for only in-use reaction wheels. The resulting state space is represented as

$$
\begin{aligned}
&S = \{s_1, s_2, ..., s_{N5}\} \\
&\textit{where} \\
&s_i = \{Bl_i, Bo_i, V_i\} \\
&Bl_i = \{bl_i^1, ..., bl_i^3\}, bl_i^j \in \{0,1\}, Bo_i = \{bo_i^1, ..., bo_i^3\}, bo_i^j \in \{0,1\} \\
&V_i = \{v_i^1, ..., v_i^7\}, v_i^j \in \{1,2,3\}
\end{aligned}
\tag{7.3.9}
$$

The thresholds in Equation (7.3.9) are arranged according to the following table. Note that there is no threshold corresponding to the no-fault scenario because it isn't modeled.

Table 7-3: Threshold variables and their relevant fault scenarios

Threshold Variable	Relevant Fault Scenario
v_1	[1 0 0]: Failure of reaction wheel 1
v_2	[0 1 0]: Failure of reaction wheel 2
v_3	[0 0 1]: Failure of reaction wheel 3
v_4	[1 1 0]: Failure of reaction wheels 1 and 2
v_5	[0 1 1] : Failure of reaction wheels 2 and 3
v_6	[1 0 1] : Failure of reaction wheels 1 and 3
v_7	[1 1 1] : Failure of all three reaction wheels

The size of the state space for this MDP is 139,968 which is manageable. As shown there are seven thresholds corresponding to the seven possible fault cases, i.e. three single wheel failure cases, three dual-wheel failure cases, and one failure case where all 3 remaining wheels fail.

The actions for this MDP are defined as,

$$M = \{\mu_{1+}, \mu_{1-}, ..., \mu_{7+}, \mu_{7-}, NOOP\} \tag{7.3.10}$$

where each of the first 14 actions is used to increase or decrease the associated conflict resolution threshold and the last action is for no operation. As was described in Section 5.10.2 the reward function for this conflict resolution MDP is defined as

$$R(s_i) = \lambda e^{-J(s_i)}, J(s_i) = \left\{ \begin{array}{c} q_1 \sum_{k=1}^{3} \{P(FA_k \mid V_i) + P(MD_k \mid V_i)\} \\ + \sum_{k1,k2 \in \{1,2,3\}} q_2 \left| bl_i^{k1} - bo_i^{k2} \right| \end{array} \right\} \tag{7.3.11}$$

Again, q_1 = q_2 = 1 and the probabilities of false alarms and missed detection are calculated based on the following formula:

$$P\left(WrongFaultDecision_k \mid V\right) = \frac{\sum_{v \in V^k} 0.3\left((v == 1) \,||\, (v == 3)\right) + 0.1(v == 2)}{Cardinality(V^k)}$$

$$P\left(WrongFaultDecision_k \mid V\right) = \begin{cases} P(FA_k \mid V) & if: bo_k = 1 \\ P(MD_k \mid V) & f: bo_k = 0 \end{cases} \tag{7.3.12}$$

Here, V^k is the set of thresholds related to fault k. Note that the above formula calculates the probability of false alarm if the corresponding fault flag is set to 1. Otherwise, the formula calculates the probability of missed detection.

To understand the transition probabilities, we summarize the process of observer-based fault detection. First each fault scenario is assigned a transition threshold (except for the no-fault scenario assumed to be the initial state). When the estimated states from the

actual system are compared with the predicted states corresponding to a particular fault scenario, a residual signal is generated. This residual signal estimates the difference between the actual states and the states for the corresponding fault scenario. If the difference/residual is smaller than the threshold for that fault scenario, the fault is identified as present. For cases where more than one scenario is likely, the tie can be broken with a default preference.

The conflict resolution MDP transition probabilities are calculated based on whether or not the state from which the threshold-change is applied has a relevant fault scenario for the changed threshold (see Table 7-3). If the scenario is relevant, increasing the threshold always results in no change and decreasing the threshold may result in either no change (with probability 0.5) or may result in a fault status change to a different state of the seven possible scenarios (with equal probabilities of 0.5/7). If on the other hand, the scenario is irrelevant, decreasing the threshold always results in no change while increasing the threshold may result in either no change (with probability 0.5) or a new fault scenario corresponding to the threshold that is increased (with probability 0.5).

7.3.3 Control Reconfiguration MDP

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

$$
\begin{aligned}
&S = \{s_1, s_2, ..., s_{N6}\} \\
&s_i = \{A_i, B_i, F_i, O_i, sw_i, c_i\} \\
&\textit{where}, \\
&A_i \in \{0,1,2\}, B_i = \{b_i^1, b_i^2, ..., b_i^5\}, b_i^j \in \{0,1\} \\
&F_i = \{f_i^1, ..., f_i^{10}\}, f_i^j \in \{0,1\}, O_i = \{o_i^1, ..., o_i^{10}\}, o_i^j \in \{0,1\} \\
&sw_i = \{sw_i^{roll}, sw_i^{pitch}, sw_i^{yaw}\}, sw_i^j \in \{0,1\}, c_i \in \{1,2,...,14\}
\end{aligned}
\tag{7.3.13}
$$

In the above equation, F represents conflict-resolved fault flags generated by the fault detection schemes for the six gyroscopes and four reaction wheels. Note that we have used a set of unified fault flags here instead of separate flags i.e. *Bl* and *Bo*. If there are any conflicts in *Bl* and *Bo* that are not resolved by the conflict resolution MDP (since the conflict resolution MDP does not always guarantee the resolution of every conflict, see Section 5.13), maximum likelihood with default preference of one flag type over the other for tie breaking is used to generate the set of fault flags in F. 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. Note that the fault flag for the reaction wheel that is not in use is stored from the last time when it was in use. Also, if a reaction wheel has never been used, it is assumed to be healthy and its fault flag has value zero.

With the above definition of state, the total number of states for the MDP is 11,274,289,152 which is too large to handle easily in an integrated MDP. 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. Also we

assume that the control law uses gyroscope readings only when the fault flags for gyroscopes are turned off. The reconfiguration of gyroscopes can therefore be separated from the reconfiguration of the control laws. The subsystem for control law reconfiguration then becomes

$$
\begin{aligned}
&S = \{s_1, s_2, ..., s_{N6}\} \\
&s_i = \{A_i, B_i, F_i, O_i, c_i\} \\
&\textit{where}, \\
&A_i \in \{0,1,2\}, B_i = \{b_i^1, b_i^2, ..., b_i^5\}, b_i^j \in \{0,1\} \\
&F_i = \{f_i^1, ..., f_i^4\}, f_i^j \in \{0,1\}, O_i = \{o_i^1, ..., o_i^4\}, o_i^j \in \{0,1\} \\
&c_i \in \{1,2,...,14\}
\end{aligned}
\tag{7.3.14}
$$

where A and B are the same as in Equation (7.3.1). Vector F contains fault flags for the four reaction wheels with axis order yaw, pitch, roll, and skew. Vector O contains the sensor-based values of the operational modes for the four wheels in the same order as in F. Variable c represents the active control law for the 14 possible fault scenarios. Note that the scenario where all four reaction wheels fail is not considered here. The size of this state space is 344,064 which is manageable. The states for gyroscope reconfiguration are then defined as

$$
\begin{aligned}
&S = \{s_1, s_2, ..., s_{N8}\} \\
&s_i = \{F_i, O_i, sw_i\} \\
&\textit{where}, \\
&F_i = \{f_i^1, ..., f_i^6\}, f_i^j \in \{0,1\}, O_i = \{o_i^1, ..., o_i^6\}, o_i^j \in \{0,1\} \\
&sw_i = \{sw_i^{roll}, sw_i^{pitch}, sw_i^{yaw}\}, sw_i^j \in \{0,1\}
\end{aligned}
\tag{7.3.15}
$$

The flags in vectors F and O in Equation (7.3.15) are arranged such that the first three flags correspond to the gyroscopes in the IRU-A set and the next three flags correspond

to the gyroscopes in the IRU-B set. The axes are in the order yaw, pitch, and roll. The size of the state-space in Equation (7.3.15) is 32,768 which is easily manageable. Variable *sw* represents the current selection of gyroscopes among IRU-A and IRU-B along each axis. Note that we assume here that along any axis, the gyroscope can be selected from any of the two sets. 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 these selections are deterministic; therefore, there are no transition probabilities involved. The reward function for the control law reconfiguration MDP is defined as

$$R(s_i) = \lambda \exp\left\{\begin{array}{l} -\sum_{k=1}^{5} \beta_k IsMissionCritical(c_i, b_i^k) - \alpha IsActionCritical(c_i, A_i) \\ -G(F_i, O_i, c_i) \end{array}\right\}$$

where,

$$IsMissionCritical(c_i, b_i^k) = \begin{cases} 0 & if: b_i^k = 1, or, c_i \in \{1,2,3,4\} \\ 1 & if: b_i^k = 0, and, c_i \in \{5,6,..10\} \\ 2 & if: b_i^k = 0, and, c_i \in \{11,12,13,14\} \end{cases}$$

$$IsActionCritical(c_i, A_i) = \begin{cases} 0 & if: A_i = 0, or, c_i \in \{1,2,3,4\} \\ 1 & if: A_i > 0, and, c_i \in \{5,6,..10\} \\ 2 & if: A_i > 0, and, c_i \in \{11,12,13,14\} \end{cases}$$

(7.3.16)

In the above equation, the *IsMissionCritical* function determines the penalty of using a 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 $\beta_k = 1$ for all k except for $\beta_3 = 5$ to indicate that the third objective 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 $\alpha = 2$ and define *IsActionCritical* to return 0 if either $A = 0$ or *c* uses 3 wheels, 1 if $A > 0$ and *c* uses two wheels, and 2 if $A > 0$ and *c* uses 1 wheel. *G* penalizes control laws 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* deactivated. These penalties are summed 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 wheels 1, 2, and 3 would be 8. However, if *c* uses three wheels and all of them are deemed healthy by both *F* and *O*, then there is no penalty for not using the fourth wheel. Details of the *G* function for each control law is presented in Table 7-4.

Table 7-4: Calculation of *G*(*F*, *O*, *c*) function

In use Control law	**Function for calculating value of *G*** *F*(i) = 1 if ith wheel is faulty *O*(i) = 0 if ith wheel is faulty
c = 1, wheel(s) in use {1, 2, 3}	`(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)))`
c = 2, wheel(s) in use {1, 2, 4}	`(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)))`
c = 3, wheel(s) in use {1, 3, 4}	`(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)))`
c = 4, wheel(s) in use {2, 3, 4}	`(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)))`
c = 5, wheel(s) in use {1, 2}	`(2.5*(F(1)+F(2)) + (~O(1)+~O(2)) + (~F(3)*2.5+~F(4)*2.5) + (O(3)+O(4)))`
c = 6, wheel(s) in	`(2.5*(F(1)+F(3)) + (~O(1)+~O(3)) + (~F(2)*2.5+~F(4)*2.5) + (O(2)+O(4)))`

use {1, 3}	
c = 7, wheel(s) in use {1, 4}	(2.5*(F(1)+F(4)) + (~O(1)+~O(4)) + (~F(3)*2.5+~F(2)*2.5) + (O(3)+O(2)))
c = 8, wheel(s) in use {2, 3}	(2.5*(F(2)+F(3)) + (~O(2)+~O(3)) + (~F(1)*2.5+~F(4)*2.5) + (O(1)+O(4)))
c = 9, wheel(s) in use {2, 4}	(2.5*(F(2)+F(4)) + (~O(2)+~O(4)) + (~F(3)*2.5+~F(1)*2.5) + (O(3)+O(1)))
c = 10, wheel(s) in use {3, 4}	(2.5*(F(3)+F(4)) + (~O(3)+~O(4)) + (~F(1)*2.5+~F(2)*2.5) + (O(1)+O(2)))
c = 11, wheel(s) in use {1}	(2.5*(F(1)) + (~O(1)) + (~F(3)*2.5+~F(4)*2.5+~F(2)*2.5) + (O(3)+O(4)+O(2)))
c = 12, wheel(s) in use {2}	(2.5*(F(2)) + (~O(2)) + (~F(3)*2.5+~F(4)*2.5+~F(1)*2.5) + (O(3)+O(4)+O(1)))
c = 13, wheel(s) in use {3}	(2.5*(F(3)) + (~O(3)) + (~F(1)*2.5+~F(4)*2.5+~F(2)*2.5) + (O(1)+O(4)+O(2)))
c = 14, wheel(s) in use {4}	(2.5*(F(4)) + (~O(4)) + (~F(3)*2.5+~F(1)*2.5+~F(2)*2.5) + (O(3)+O(1)+O(2)))

The reward function for the reconfiguration of gyroscopes can be defined with only the G term similar to the reward function for the control law reconfiguration. Note that in this model there are no constraints in the reconfiguration options and no uncertainties in the state transitions. Therefore, the policy for this MDP is also calculated using greedy search instead of value iteration.

7.3.4 CFT-SOAP Execution

After having formulated all the MDPs related to planning, fault detection, and reconfiguration, in this section we describe the real-time execution of all the MDP policies onboard the spacecraft. Figure 7.3 represents the block diagram for CFT-SOAP

execution for the FUSE mission. This figure is similar to Figure 3.3 except for the fact that instead of one integrated MDP policy, we have separate policies. In Figure 7.3, all of the information from the spacecraft sensors, observer-based fault detector, system dynamics, and system configuration is obtained by the real-time executive. The current state for each of the MDPs on the left side of the figure is extracted from this information. Note that extracting the current state for the logic-based fault detection MDP involves determining which of the wheels and gyroscopes are currently in use. This can be easily done using the current gyroscope switching and control law configuration. Also the execution of the current state for reconfiguration MDPs involves generation of the F vector that may require resolving the unresolved fault conflicts from the previous time step using the maximum likelihood method. Once the state information for all the MDPs is generated, the corresponding updated actions are extracted from the MDP policies. These actions are then converted into commands and are sent to the system configuration manager, reconfigurable controller, and observer-based fault detector. The interpretation of the actions here includes the selection of the actions among $\mu_{\text{P-1}}$ and $\mu_{\text{P-2}}$ based on the fault information. If the fault information indicates full control availability, $\mu_{\text{P-1}}$ is executed; otherwise $\mu_{\text{P-2}}$ is executed.

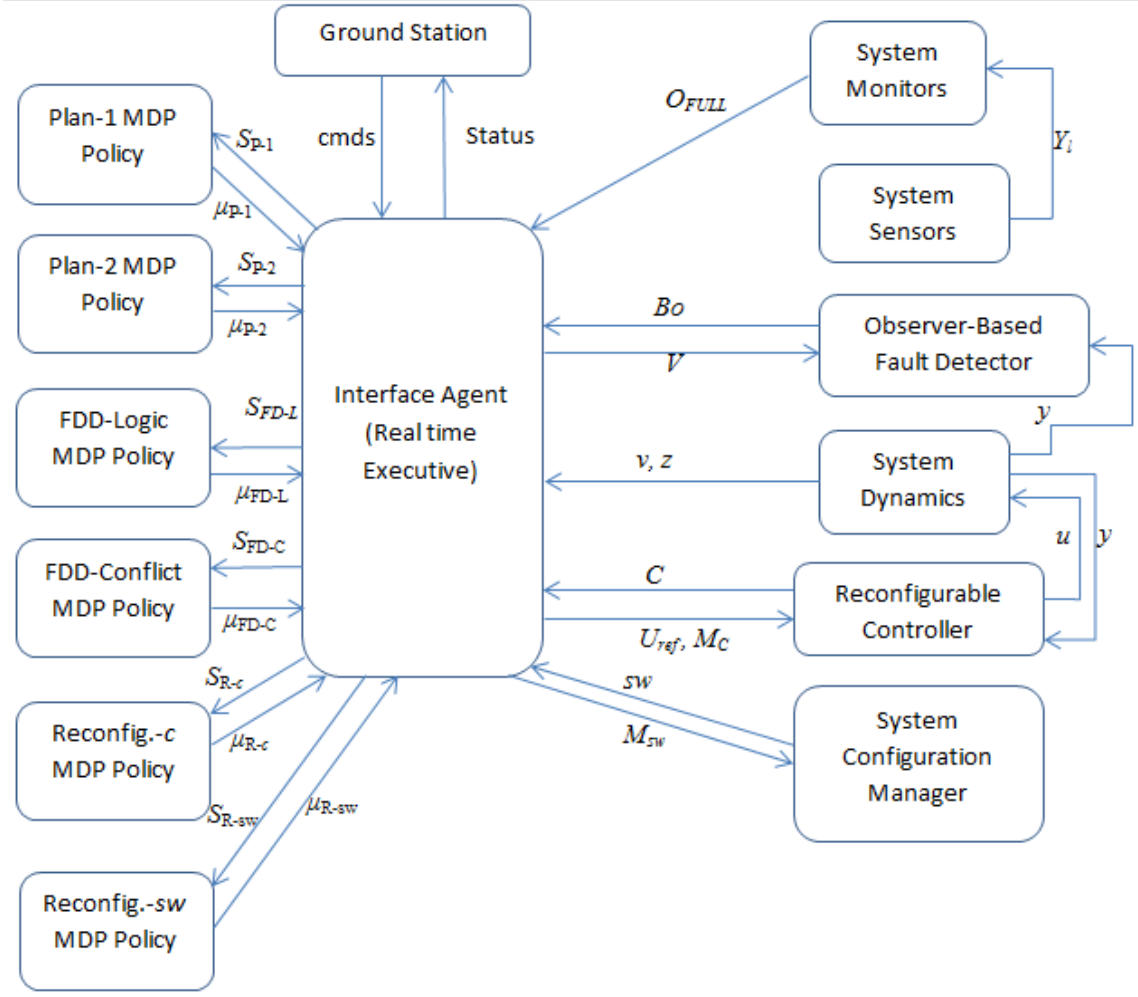

Figure 7.3: Online execution of CFT-SOAP for the FUSE mission

7.4 FUSE Modeling with ASPEN and Livingstone

ASPEN and Livingstone formulations generate plans and post-fault sequences of activities, respectively. An ASPEN model [85] has seven basic components: 1) Parameters, 2) Parameter dependencies, 3) Temporal constraints, 4) Resources, 5) State variables, 6) Activities, 7) Activity reservations. Once a planning problem is modeled in the ASPEN framework, the solution (plan) is computed by using iterative repair search that may be guided by appropriate heuristics [22]. Conceptually, iterative repair search starts with an approximate plan supplied by the user that may have conflicts and that may

not be complete. The conflicts in the initial plan are resolved one by one by adding, deleting, and adjusting activities. Sometimes larger activities may be decomposed into primitive actions while at other instances, actions may be aggregated. The ASPEN model can be changed online as a result of faults. For example new constraints might be imposed on the activities. Under such situations, the plan is repaired online using the updated model and iterative repair search.

Livingstone [104], on the other hand, is an online reactive deduction system that serves the purposes of fault detection and reconfiguration. The basic modeling entity in Livingstone is a state transition system. A state transition system is composed of three components: 1) State variables, 2) The domain space for state variables, 3) The set of possible state transitions. Each component of the spacecraft can be represented as a state transition system. Together all state transition system layers form the plant transition system. The problem of fault detection is addressed by using combinatorial optimization over design variables X, constraints Y, and objective function W. Each variable in X represents a component in the plant whose values are the possible component transitions. Each possible plant transition corresponds to assignment of values to the variables in X. Y is the constraint that all state transitions should belong to the set of allowable transitions. The objective function is the probability of each plant transition. For FUSE, a transition might include any number of components transitioning from normal to a faulty state and hence the objective function would be the probability of occurrence of the corresponding faults given the sensor observations. Specifics of ASPEN and Livingstone models for application to the FUSE case study are provided below.

7.4.1 The ASPEN Model

The ASPEN model translation for the FUSE mission is represented in Table 7-5. There are a total of 15 activities, five of which include data collection activities while the remaining 10 are attitude maneuver activities. Each activity has associated constraints and parameters. Although there are five types of constraints listed in Table 7-5, not all activities have all five constraints active. For example the constraint of the data collection window is active only for activities of type *data collection*.

Table 7-5: ASPEN model

Model Attribute	Description	Remarks
Activities	Five data collection activities, one for each science target Ten attitude-maneuver activities, one maneuver between any two different target pointings.	Each activity has its own parameters such as energy required, sun avoidance, instruments required, feasible true anomaly window, etc
Constraints	Data collection window Sun exposure Energy Depletion Attitude Pointing Instrument Health	Constraints are matched with the activity parameters to make sure that the activity is feasible.
States	True Anomaly Orbit count Attitude Pointing Data Collection Flags Available Energy Eclipse-Sun Flag Available Instruments	States are obtained from various sensors and fault detection schemes in the system.
Activity Parameters	Feasible True anomaly Window Sun Avoidance flag Energy Required Attitude Pointing Required Instruments Required Change in True Anomaly Incurred	Attitude pointing activities do not have a true anomaly window requirement or sun avoidance requirement

The state information reflects the situation of the spacecraft related to its mission. Note that the state variables are the same as for the planning MDP in the CFT-SOAP formulation of FUSE. Symbolically, the above model can be represented as:

$$
\begin{aligned}
&ASPEN = \{A, C, S, P\} \\
&A = \{a_1, a_2, ..., a_{15}\} \\
&C = \{c_{1,1}, c_{1,2}, ..., c_{1,5}, c_{2,1}, ..., c_{15,5}\} \\
&S = \{s_1, ..., s_7\} \\
&P = \{p_{1,1}, p_{1,2}, ..., p_{1,5}, p_{2,1}, ..., p_{5,5}, p_{6,1}, ..., p_{6,4}, p_{7,1}, ..., p_{15,4}\}
\end{aligned} \tag{7.4.1}
$$

Each activity in ASPEN can be assigned a true anomaly range and an orbit count number. No two activities are allowed to overlap. There are five possible constraints for each activity assignment. Each binary element of the constraint set $c_{i,j}$ represents the presence or absence of constraint j in activity i. There are seven state variables each with a respective domain. For example, the domain of true anomaly state variable ranges from 0 to 360 degrees in discrete intervals. The domain of orbit count is the set of natural numbers up to a predefined maximum limit. The domain of the attitude pointing variable is the set of the first five natural numbers. Note that the domains of the state variable should be finite; otherwise there is a possibility of never being able to find a feasible plan. The goal state in our case study is any state where the data collection flag variable indicates all of the required data has been collected.

The ASPEN planning process starts with an initial state and a set of randomly-assigned activities. If there are no constraints and the goal state is reached at the end of the last activity, the solution is found. Otherwise, activities are added, deleted, or

adjusted/reassigned until all constraints are satisfied and the goal state is depicted to be achieved by the end of the last activity in the plan.

7.4.2 Livingstone Model

For the Livingstone model, each of the four reaction wheels and six gyroscopes is represented as a state transition system where state represents the health status in terms of *failed*/*normal*. There are four possible transitions for each of the state transition systems i.e. *normal* to *failed*, *normal* to *normal*, *failed* to *normal*, and *failed* to *failed*. Therefore, the overall spacecraft health transition is a collection of transitions of each of the ten components of interest for our case study. The combinatorial optimization problem therefore, is presented as:

$$
\begin{aligned}
&CO_{FD} = \{X, Y, W\} \\
&where, \\
&X = \{x_1, \ldots, x_{10}\} \\
&Y = \{y_1, \ldots, y_{10}\} \\
&W = \{w_1, \ldots, w_{10}\} \\
&w_i = x_i P(x_i \mid y_i) + (1 - x_i) P(1 - x_i \mid y_i)
\end{aligned} \tag{7.4.2}
$$

The above equation is based on the assumption of the independence of faults. Therefore, failure detection of the i^{th} component can be optimized with respect to objective function w_i. w_i is a function of x_i and has maximum value for the most consistent value of x_i given y_i, where the most consistent refers to the value of x_i which is most likely based on the available information. Binary flag x_i represents the *normal/faulty* mode of the i^{th} component. Binary flag y_i represents the observed mode of the i^{th} component. There are no constraints in mode transitions of the components, i.e., all four possible transitions are allowed.

For reconfiguration, the combinatorial optimization problem selects among available reconfiguration options such that the current plan is executable. The objective function is set to Equation (7.3.16) to maximize consistency between the Livingstone and CFT-SOAP models. Variables in X are possible reconfigurations, Y includes flags for infeasible reconfigurations, and W is a function of X and Y whose value can be maximized with the selection of the best among the feasible values of variables in X.

$$
\begin{aligned}
&CO_{RC} = \{X, Y, W\} \\
&where, \\
&X = \{x_1, x_2\} \\
&Y = \{y_1, \ldots, y_{14}\} \\
&W = \{w_1(x_1), w_2(x_2, Y)\}
\end{aligned}
\tag{7.4.3}
$$

In the above equation, x_1 is used to select a combination of gyroscopes and x_2 is used to select a combination of reaction wheels. Variables in Y indicate feasible/infeasible status for each combination of the reaction wheels. This status is calculated based on the *Instruments Required* parameter of the remaining plan activities e.g. if any of the remaining activities of the current plan generated by ASPEN require three reaction wheels, all the combinations of the reaction wheels where less than three wheels are in use are marked *infeasible*. Variables in W calculate the cost of using a particular configuration that is similar to the G function in Equation (7.3.16).

7.4.3 Execution of ASPEN-Livingstone

In this section we present an online execution scheme for the ASPEN-Livingstone framework. Figure 7.4 shows the online execution scheme where the real-time execution module collects the system health and system output (angular velocities and attitude)

information from the system monitors and the real-time control scheme respectively. This information is used to generate the initial state for ASPEN to request a new plan if there is no in-progress plan or the in-progress plan is no longer applicable due to a fault. Once the plan is acquired, the real-time executive executes the plan activities one by one by issuing real-time control commands. Meanwhile the real-time executive also monitors the health of the system using information from the system monitors that use information from the sensors to generate operational modes for the components that represent component health. The observed modes for the components are sent to the mode identification module that uses the Livingstone fault detection scheme to determine the fault flags for the reaction wheels and gyroscopes. If there is a new fault, the reconfiguration request is sent to the Livingstone-based reconfiguration module along with fault information and information about the current plan. The reconfiguration module either sends a corresponding reconfiguration command if possible; otherwise a new plan request must be sent to the ASPEN by the real-time executive.

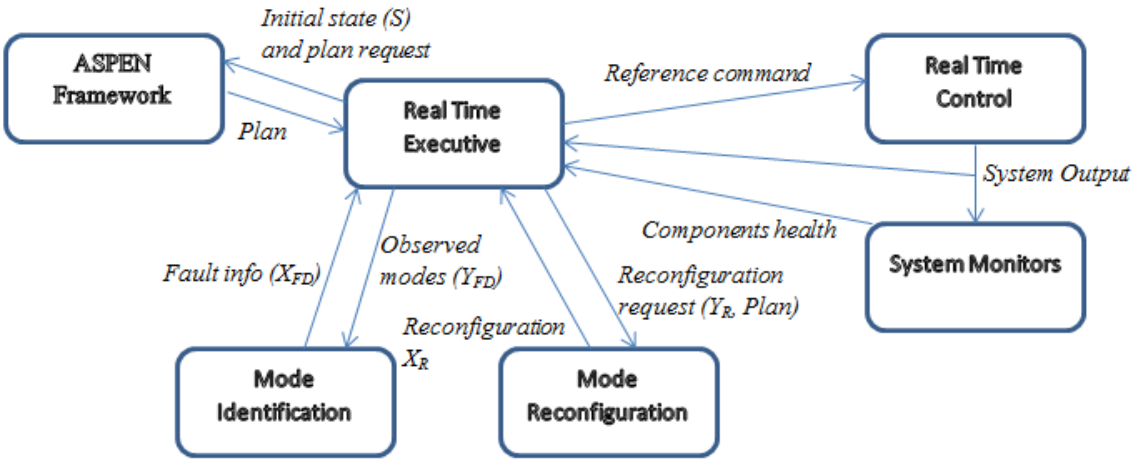

Figure 7.4: ASPEN-Livingstone execution for the FUSE mission

7.5 Simulation Results

In this section, we simulate the sequence of failures from the actual FUSE mission and report the responses from CFT-SOAP and ASPEN-Livingstone models. Table 7-6 lists the gyroscope and reaction wheel failure cases actually encountered during the FUSE mission (as of 2006) [91].

Table 7-6: Status of the FUSE gyroscopes and reaction wheels as of 2006

Axis	IRU-A	IRU-B	Reaction Wheel Assemblies
Yaw	1/6/00 Warning flag tripped **Operational**	12/10/02 Warning flag tripped 7/31/03 Failed	2/16/01 Stopped; restarted in 11 days 11/25/01 Failed
Pitch	1/18/00 Warning flag tripped **Operational**	8/31/01 Warning flag tripped 9/28/04 Noisy / Turned off	8/04/00 Stopped; restarted in 40 days 12/10/01 Failed
Roll	4/19/00 Warning flag tripped 5/30/01 Failed	10/06/01 Warning flag tripped 5/17/05 Failed	12/17/03 Stopped; restarted in 2 hrs 12/27/04 Failed
Skew	----------	----------	**Operational**

Although the failures were separated from each other by months or in some cases years, we compress the failures into the course of a few orbits to facilitate their integration into the case study. This is a worst-case scenario, and supports comparison of the capabilities of CFT-SOAP, ASPEN, and Livingstone to support reactive mission configuration without assistance from the ground station.

7.5.1 Simulation with CFT-SOAP

7.5.1.1 Trajectory Analysis

MDP policies for planning (with full control availability) as well as fault detection and control reconfiguration were calculated using MATLAB software with an Intel core i5 processor. Table 7-7 shows the computation times for the MDPs. Note that all MDPs have been solved using a finite horizon. The discount factor (γ) and epoch for the planning MDP have higher values. The discount factor has a high value because a plan needs to consider values of future rewards. The epoch has value 20 to make sure that the mission is completed. The discount factors and epochs for the fault detection and reconfiguration MDPs are low because these MDPs require emphasis of more immediate actions to resolve the fault scenario.

Table 7-7: MDP computation times

MDP type	Computation time (sec)	Discount factor (γ)	Epoch
Planning (full control authority)	103.72	0.99	20
Logic-based fault detection	5.9	0.5	3
Conflict resolution	33.85	0.5	3
Control law reconfiguration	88.44	0.5	3
Gyroscope reconfiguration	5.9	0.5	3

Online execution is performed as depicted in Chapter 3 (Figures 3.3. and 3.4). Policies generated for FUSE are shown in Table 7-8. The fault flags in *Bl*, *Bo*, and *F* as well as the flags in *O* are arranged such that the first three flags correspond to the IRU-A set of gyroscopes in the order of yaw, pitch and roll axes. The next three flags correspond to the IRU-B set of gyroscopes in the same order of axes, and the final four flags correspond to the reaction wheel assemblies in the order of yaw, pitch, roll, and skew axes. The

thresholds and control laws are ordered as in Table 7-4 where wheel 1 corresponds to the yaw axis, wheel 2 corresponds to the pitch axis, wheel 3 corresponds to the roll axis, and wheel 4 corresponds to the skew axis.

We start with a normal state in step 1 with attitude pointing at Target 1 and true anomaly value of 1. In the response, there is no reconfiguration required from the control law or gyroscope reconfiguration policy. The fault detection policy does not detect any fault and the planning MDP suggests $NOOP_{50}$. Before the start of step 2, the spacecraft moves by 50 degrees of true anomaly and meanwhile the observation flags for the yaw and pitch axes gyroscopes in IR-A unit turn off indicating potential failures. In step 2, the spacecraft is pointed towards Target 1 and is within the window of visibility therefore the planning MDP suggests μ_1 i.e. to collect data from Target 1. The reconfiguration policy is to switch the yaw and pitch axis gyroscopes from IRU-A to IRU-B. No more failures occur at this point. Before step 3, data is collected from Target 1 and the spacecraft advances by 10 degrees of true anomaly. Also, the reconfiguration of gyroscopes commanded by the reconfiguration policy is carried out while two more observation flags turn off, one for the roll axis gyroscope in IRU-A and one for Reaction Wheel 2. In step 3, the reconfiguration policies change the control law and gyroscope configurations to avoid usage of potentially faulty components. Similarly, in the remaining steps, the response of the MDP policies to the sequence of failures is presented. By the end of step 11, all scientific data is collected, the spacecraft has advanced to the true anomaly value of 261, the spacecraft is in an eclipse condition that started at the end of step 11, and the energy left in the batteries is 2 units. By the end of step 13, only one of the four reaction wheels is working, and only two of the six gyroscopes are operational.

Table 7-8: Simulation case study for CFT-SOAP with FUSE

Sim Step #	State of the FUSE Spacecraft	MDP Policies	Exogenous and Policy-related Events
1	$A = 0$, $B = \{0, 0, 0, 0, 0\}$ $v = 1$, $z = 1$, $SE = 3$, $EL = 5$ $Bl = \{0,0,0,0,0,0,0,0,0,0\}$ $Bo = \{0,0,0,0,0,0,0,0,0,0\}$ $F = \{0,0,0,0,0,0,0,0,0,0\}$ $O = \{1,1,1,1,1,1,1,1,1,1\}$ $V = \{2,2,2,2,2,2,2,2,2,2,2,2,2,2\}$ $sw = \{1, 1, 1\}$, $c = 1$; wheels $\{1, 2, 3\}$	$\pi_{Plan, full} = NOOP_{50}$ $\pi_{Plan, partial}$ = Not in use $\pi_{FDD, Logic}$ = NOOP $\pi_{FDD, Conflict}$ = NOOP $\pi_{Reconfig, gyro}$ = NOOP $\pi_{Reconfig, Control}$ = NOOP	$v = 51$ O = {0,0,1,1,1,1,1,1,1,1}
2	$A = 0$, $B = \{0, 0, 0, 0, 0\}$ $v = 51$, $z = 1$, $SE = 3$, $EL = 5$ $Bl = \{0,0,0,0,0,0,0,0,0,0\}$ $Bo = \{0,0,0,0,0,0,0,0,0,0\}$ $F = \{0,0,0,0,0,0,0,0,0,0\}$ $O = \{0,0,1,1,1,1,1,1,1,1\}$ $V = \{2,2,2,2,2,2,2,2,2,2,2,2,2,2\}$ $sw = \{1, 1, 1\}$, $c = 1$; wheels $\{1, 2, 3\}$	$\pi_{Plan, full} = \mu_1$ $\pi_{Plan, partial}$ = Not in use $\pi_{FDD, Logic}$ = NOOP $\pi_{FDD, Conflict}$ = NOOP $\pi_{Reconfig, gyro} = \{2,2,1\}$ $\pi_{Reconfig, Control}$ = NOOP	$A = 2$, $A = 0$ $B = \{1, 0, 0, 0, 0\}$, $v = 61$ $O = \{0,0,0,1,1,1,1,0,1,1\}$ $sw = \{2,2,1\}$
3	$A = 0$, $B = \{1, 0, 0, 0, 0\}$ $v = 61$, $z = 1$, $SE = 3$, $EL = 5$ $Bl = \{0,0,0,0,0,0,0,0,0,0\}$ $Bo = \{0,0,0,0,0,0,0,0,0,0\}$ $F = \{0,0,0,0,0,0,0,0,0,0\}$ $O = \{0,0,0,1,1,1,1,0,1,1\}$ $V = \{2,2,2,2,2,2,2,2,2,2,2,2,2,2\}$ $sw = \{2, 2, 1\}$, $c = 1$; wheels $\{1, 2, 3\}$	$\pi_{Plan, full} = \mu_2$ $\pi_{Plan, partial}$ = Not in use $\pi_{FDD, Logic}$ = NOOP $\pi_{FDD, Conflict}$ = NOOP $\pi_{Reconfig, gyro} = \{2,2,2\}$ $\pi_{Reconfig, Control} = 3$	$A = 1$, $A = 0$ $v = 81$, $z = 2$ $c = 3$; wheels $\{1, 3, 4\}$ $sw = \{2,2,2\}$ $O = \{0,0,0,1,0,0,0,0,1,1\}$
4	$A = 0$, $B = \{1, 0, 0, 0, 0\}$ $v = 81$, $z = 2$, $SE = 3$, $EL = 5$ $Bl = \{0,0,0,0,0,0,0,0,0,0\}$ $Bo = \{0,0,0,0,0,0,0,0,0,0\}$ $F = \{0,0,0,0,0,0,0,0,0,0\}$ $O = \{0,0,0,1,0,0,0,0,1,1\}$ $V = \{2,2,2,2,2,2,2,2,2,2,2,2,2,2\}$ $sw = \{2, 2, 2\}$, $c = 3$; wheels $\{1, 3, 4\}$	$\pi_{Plan, full} = NOOP_{20}$ $\pi_{Plan, partial}$ = Not in use $\pi_{FDD, Logic} = Bl(7)$ $\pi_{FDD, Conflict}$ = NOOP $\pi_{Reconfig, gyro}$ = NOOP $\pi_{Reconfig, Control}$ = NOOP	$v = 101$ $Bl = \{0,0,0,0,0,0,1,0,0,0\}$ $Bo = Bl = F$
5	$A = 0$, $B = \{1, 0, 0, 0, 0\}$	$\pi_{Plan, full} = \mu_2$	$A = 2$, $A = 0$

	v = 101, z = 2, SE = 3, EL = 5 Bl = {0,0,0,0,0,0,1,0,0,0} Bo = {0,0,0,0,0,0,1,0,0,0} F = {0,0,0,0,0,0,1,0,0,0} O = {0,0,0,1,0,0,0,0,1,1} V = {2,2,2,2,2,2,2,2,2,2,2,2,2,2,2} sw = {2, 2, 2}, c = 3; wheels {1, 3, 4}	$\pi_{\text{Plan, partial}}$ = Not in use $\pi_{\text{FDD, Logic}}$ = Bl(8) $\pi_{\text{FDD, Conflict}}$ = NOOP $\pi_{\text{Reconfig, gyro}}$ = NOOP $\pi_{\text{Reconfig, Control}}$ = 4	v = 111 B = {1, 1, 0, 0, 0} Bl = {0,0,0,0,0,0,1,1,0,0} Bo = Bl = F c = 4; wheels {2, 3, 4}
6	A = 0, B = {1, 1, 0, 0, 0} v = 111, z = 2, SE = 3, EL = 5 Bl = {0,0,0,0,0,0,1,1,0,0} Bo = {0,0,0,0,0,0,1,1,0,0} F = {0,0,0,0,0,0,1,1,0,0} O = {0,0,0,1,0,0,0,0,1,1} V = {2,2,2,2,2,2,2,2,2,2,2,2,2,2,2} sw = {2, 2, 2}, c = 3; wheels {2, 3, 4}	$\pi_{\text{Plan, full}}$ = μ_3 $\pi_{\text{Plan, partial}}$ = Not in use $\pi_{\text{FDD, Logic}}$ = Bl(4) $\pi_{\text{FDD, Conflict}}$ = NOOP $\pi_{\text{Reconfig, gyro}}$ = NOOP $\pi_{\text{Reconfig, Control}}$ = 10	A = 1, A = 0 v = 161, z = 3 O = {0,0,0,0,0,0,0,0,1,1} Bl = {0,0,0,1,0,0,1,1,0,0} Bo = Bl = F c = 10; wheels {3, 4}
7	A = 0, B = {1, 1, 0, 0, 0} v = 161, z = 3, SE = 3, EL = 5 Bl = {0,0,0,1,0,0,1,1,0,0} Bo = {0,0,0,1,0,0,1,1,0,0} F = {0,0,0,1,0,0,1,1,0,0} O = {0,0,0,0,0,0,0,0,1,1} V = {2,2,2,2,2,2,2,2,2,2,2,2,2,2,2} sw = {2, 2, 2}, c = 10; wheels {3, 4}	$\pi_{\text{Plan, full}}$ = μ_3 $\pi_{\text{Plan, partial}}$ = Not in use $\pi_{\text{FDD, Logic}}$ = NOOP $\pi_{\text{FDD, Conflict}}$ = NOOP $\pi_{\text{Reconfig, gyro}}$ = {1,2,2} $\pi_{\text{Reconfig, Control}}$ = NOOP	A = 2, A = 0 v = 171 B = {1, 1, 1, 0, 0} O = {0,0,0,0,0,0,0,0,0,1} sw = {1,2,2}
8	A = 0, B = {1, 1, 1, 0, 0} v = 171, z = 3, SE = 3, EL = 5 Bl = {0,0,0,1,0,0,1,1,0,0} Bo = {0,0,0,1,0,0,1,1,0,0} F = {0,0,0,1,0,0,1,1,0,0} O = {0,0,0,0,0,0,0,0,0,1} V = {2,2,2,2,2,2,2,2,2,2,2,2,2,2,2} sw = {1, 2, 2}, c = 10; wheels {3, 4}	$\pi_{\text{Plan, full}}$ = μ_4 $\pi_{\text{Plan, partial}}$ = Not in use $\pi_{\text{FDD, Logic}}$ = Bl(5) $\pi_{\text{FDD, Conflict}}$ = NOOP $\pi_{\text{Reconfig, gyro}}$ = NOOP $\pi_{\text{Reconfig, Control}}$ = NOOP	A = 1, A = 0 v = 221, z = 4 Bl = {0,0,0,1,1,0,1,1,0,0} Bo = Bl = F
9	A = 0, B = {1, 1, 1, 0, 0} v = 221, z = 4, SE = 3, EL = 5 Bl = {0,0,0,1,1,0,1,1,0,0} Bo = {0,0,0,1,1,0,1,1,0,0} F = {0,0,0,1,1,0,1,1,0,0} O = {0,0,0,0,0,0,0,0,0,1} V =	$\pi_{\text{Plan, full}}$ = μ_4 $\pi_{\text{Plan, partial}}$ = Not in use $\pi_{\text{FDD, Logic}}$ = Bl(9) $\pi_{\text{FDD, Conflict}}$ = NOOP $\pi_{\text{Reconfig, gyro}}$ = {1,1,2} $\pi_{\text{Reconfig, Control}}$ =	A = 2, A = 0 v = 231 B = {1, 1, 1, 1, 0} SE = 1 Bl = {0,0,0,1,1,0,1,1,1,0} Bo = Bl = F sw = {1,1,2}

	{2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2} *sw* = {1, 2, 2}, *c* = 10; wheels {3, 4}	NOOP	
10	*A* = 0, *B* = {1, 1, 1, 0, 0} *v* = 231, *z* = 4, *SE* = 1, *EL* = 5 *Bl* = {0,0,0,1,1,0,1,1,1,0} *Bo* = {0,0,0,1,1,0,1,1,1,0} *F* = {0,0,0,1,1,0,1,1,1,0} *O* = {0,0,0,0,0,0,0,0,0,1} *V* = {2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2} *sw* = {1, 1, 2}, *c* = 10; wheels {3, 4}	$\pi_{Plan,\ full}$ = μ_5 $\pi_{Plan,\ partial}$ = Not in use $\pi_{FDD,\ Logic}$ = *Bl*(6) $\pi_{FDD,\ Conflict}$ = NOOP $\pi_{Reconfig,\ gyro}$ = NOOP $\pi_{Reconfig,\ Control}$ = 14	*A* = 1, *A* = 0 *v* = 251, *z* = 5, *EL* = 3 *Bl* = {0,0,0,1,1,1,1,1,1,0} *Bo* = *Bl* = *F* *c* = 14; wheels {4}
11	*A* = 0, *B* = {1, 1, 1, 1, 0} *v* = 251, *z* = 5, *SE* = 1, *EL* = 3 *Bl* = {0,0,0,1,1,1,1,1,1,0} *Bo* = {0,0,0,1,1,1,1,1,1,0} *F* = {0,0,0,1,1,1,1,1,1,0} *O* = {0,0,0,0,0,0,0,0,0,1} *V* = {2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2} *sw* = {1, 1, 2}, *c* = 14; wheels {4}	$\pi_{Plan,\ full}$ = Not in use $\pi_{Plan,\ partial}$ = μ_5 $\pi_{FDD,\ Logic}$ = NOOP $\pi_{FDD,\ Conflict}$ = NOOP $\pi_{Reconfig,\ gyro}$ = {1,1,1} $\pi_{Reconfig,\ Control}$ = NOOP	*A* = 2, *A* = 0 *v* = 261, *EL* = 2 B = {1, 1, 1, 1, 1} *sw* = {1,1,1}
12	*A* = 0, *B* = {1, 1, 1, 1, 1} *v* = 261, *z* = 5, *SE* = 1, *EL* = 2 *Bl* = {0,0,0,1,1,1,1,1,1,0} *Bo* = {0,0,0,1,1,1,1,1,1,0} *F* = {0,0,0,1,1,1,1,1,1,0} *O* = {0,0,0,0,0,0,0,0,0,1} *V* = {2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2} *sw* = {1, 1, 1}, *c* = 14; wheels {4}	$\pi_{Plan,\ full}$ = Not in use $\pi_{Plan,\ partial}$ = $NOOP_{10}$ $\pi_{FDD,\ Logic}$ = *Bl*(3) $\pi_{FDD,\ Conflict}$ = NOOP $\pi_{Reconfig,\ gyro}$ = {1,1,1} $\pi_{Reconfig,\ Control}$ = NOOP	*v* = 271 *Bl* = {0,0,1,1,1,1,1,1,1,0} *Bo* = *Bl* = *F*
13	*A* = 0, *B* = {1, 1, 1, 1, 1} *v* = 271, *z* = 5, *SE* = 1, *EL* = 2 *Bl* = {0,0,1,1,1,1,1,1,1,0} *Bo* = {0,0,1,1,1,1,1,1,1,0} *F* = {0,0,1,1,1,1,1,1,1,0} *O* = {0,0,0,0,0,0,0,0,0,1} *V* = {2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2} *sw* = {1, 1, 1}, *c* = 14; wheels {4}	$\pi_{Plan,\ full}$ = Not in use $\pi_{Plan,\ partial}$ = $NOOP_{10}$ $\pi_{FDD,\ Logic}$ = NOOP $\pi_{FDD,\ Conflict}$ = NOOP $\pi_{Reconfig,\ gyro}$ = NOOP $\pi_{Reconfig,\ Control}$ = NOOP	*v* = 281

7.5.1.2 Robustness Analysis

In this section, we present some results from the variation of the a priori probabilities of failures of the reaction wheels and gyroscopes. Recall that, in Section 7.3, we used the a priori failure probability values for all the reaction wheels and gyroscopes to be 0.05 and calculated the probability of transition to the failure state for the planning MDP with full control authority to be 1.25×10^{-5} times the change in true anomaly incorporated in the attitude maneuver. This value was based on a safety factor that was selected based on engineering judgement. Since engineering judgement is not always accurate, we present in this section the results of selecting inaccurate failure probabilities. In particular, we solved 10 MDPs for planning with full control availability each with a different failure probability value. Then we simulated the trajectories for each of the 10 MDPs in all of the 10 possible failure probability environments. This required 100 MDP trajectory simulations. Each of the 100 trajectory simulations was simulated for each possible initial state. Since there are 86,400 normal states (and one failure state) in each MDP (see equation 7.3.2), this amounts to 8,640,000 trajectories. To compute an average over several cases, we run each of the 8,640,000 trajectories 250 times. The total time taken by all simulation runs on Intel core i5 laptop computer was 40 hours. From all these simulations, we computed the average survival time and average expected discounted reward for each of the 10 MDPs in each of the 10 failure probability environments. The average was taken over all of the 86,400 initial states that were run 250 times.

Table 7-9 shows the indexing map for the failure probability environments used in our simulations. Table 7-10 shows the results from the simulation in the form of average survival times and average expected discounted rewards. Note that the survival time is

out of 20 steps (or a sequence of 20 actions) because the epoch (or finite horizon) of the MDP policies generated was selected to be 20 steps as shown in Table 7-7. The results indicate that the policies generated for higher probabilities of failure have a better survival rate in the environment with the highest probability of failure (environment with index 10). Also note that due to the conservative nature of the policies with higher probabilities of failure, the expected reward obtained by these policies is also lower. These results indicate a clear tradeoff between spacecraft safety and expected science reward obtained by the policy. In general, if the probabilities of failure are underestimated, the resulting policy might consist of bold decisions to collect science data that may introduce risk of failure to the spacecraft. On the other hand, if the probability of failure is overestimated, the science reward obtained from the mission may not be high. This study also suggests that if spacecraft mission planning does not take into account the probabilities of failure, the resulting plan may jeopardize spacecraft safety.

Table 7-9: Index of the MDPs with respect to the selected failure probabilities

MDP Index / Environment Index	Failure Probability
1	1.25×10^{-5}
2	2.88×10^{-4}
3	5.65×10^{-4}
4	8.41×10^{-4}
5	1.1×10^{-3}
6	1.4×10^{-3}
7	1.7×10^{-3}
8	1.9×10^{-3}
9	2.2×10^{-3}
10	2.5×10^{-3}

Table 7-10: Robustness analysis results

MDP Index	Expected reward (own environment)	Worst case expected reward	Environment index for worst case expected reward	Survival time (out of 20 steps) in own environment	Worst case survival time	Environment index for worst case survival time
1	20,720	5,909	10	19.92	12.28	10
2	16,560	6,302	10	19.19	14.19	10
3	13,780	6,662	10	19.18	17.03	10
4	11,790	6,853	10	19.35	18.46	10
5	10,360	6,955	10	19.49	19.07	10
6	9,305	7,015	10	19.56	19.33	10
7	8,503	7,047	10	19.60	19.48	10
8	7,896	7,078	10	19.68	19.63	10
9	7,452	7,108	10	19.73	19.71	10
10	7,169	7,169	10	19.75	19.75	10

7.5.2 Simulation with ASPEN-Livingstone

The following table shows the initial plan that we assume ASPEN could have generated given the initial conditions as in Table 5. The eclipse is assumed to be from 230-350 degrees of true anomaly. Note that we did not generate this plan using the actual ASPEN software but in theory, this is one of the plans that ASPEN could have generated since it is complete and satisfies all activity constraints.

Table 7-11: ASPEN plan for FUSE case study

Step Number	Plan Activity (true anomaly, orbit)
1	Data Collection 1 (50-60, 1)
2	Pointing 1-2 (70-90, 1)
3	Data Collection 2 (100-110, 1)
4	Pointing 2-3 (120-170, 1)
5	Data Collection 3 (170-180, 1)
6	Pointing 3-4 (180-230, 1)
7	Data Collection 4 (230-240, 1)
8	Pointing 4-5 (240-260, 1)
9	Data Collection 5 (260-270, 1)

Livingstone assumes the same fault sequence as in Table 5 and we also assume that all faults are correctly diagnosed. Reconfiguration decisions for Livingstone are straightforward since there is no tradeoff between mission objectives and safety of the spacecraft. All faulty components are taken out and healthy components are selected for use. In the case of partial control availability, online re-planning is required using ASPEN based on remaining available attitude maneuver activities. In short the response of ASPEN-Livingstone is similar to that of CFT-SOAP for the FUSE case study. Note that this does not mean that these two technologies are equivalent. The next section presents a comparison between the two technologies on the basis of their model expressiveness, computational complexity, and robustness of generated solutions.

7.6 Comparison between CFT-SOAP and ASPEN-Livingstone Approaches

In this section, we compare the CFT-SOAP and ASPEN-Livingstone approaches for fault tolerant planning on three bases: expressiveness of the modeling, computational complexity of the solution, and robustness to failures. Table 7-12 presents a comparison on the basis of model expressiveness. The major difference is in the expression of uncertainties. CFT-SOAP incorporates uncertainties in all three identified aspects of fault tolerant planning whereas the ASPEN-Livingstone system incorporates only the uncertainty in fault detection. The last row in Table 7-12 indicates the difference in the representation of time. Here CFT-SOAP may have a disadvantage of having discrete time but ASPEN's state-space will grow increasingly complex as temporal resolution is increased. Note that in our case study, time is represented in terms of the true anomaly in CFT-SOAP and in terms of the combination of true anomaly and orbit count in ASPEN.

Table 7-12: Comparison of model-expressiveness between CFT-SOAP and ASPEN-Livingstone modeling methods

Expressiveness Quality	CFT-SOAP	ASPEN-Livingstone
State Information	Yes	Yes
Available Decisions	Yes	Yes
Optimality Criterion	Yes	Yes: only in the Livingstone
Science-optimal Planning	Yes	No
Comprehensive Fault Tolerance	Yes	No
Transition Constraints	Yes: Implicit	Yes: Explicit
Uncertainties	Yes: Both in planning and fault tolerance	Yes: Only in Fault Detection
time	Yes: Variable length discrete time horizons	Yes: Constant length discrete time horizons

Table 7-13 presents a comparison of online and offline computational complexities. Note that the majority of computations for CFT-SOAP are performed offline. This is important because the cost of computing is much less offline (on the ground) as compared to online (onboard the spacecraft) for space missions. Also, although we have presented a nice case for ASPEN in our case study, it may not always be easy to plan and re-plan during the real missions since the time to find the plan that is sufficient for the mission is highly dependent upon amount of repairing that needs to be done and also upon the heuristics for guiding the search. In general, the search space for ASPEN is as large as all possible repair methods (Q) to all possible conflicts (C) in all possible orders. On the other hand, for fault tolerance, the online computations required by the Livingstone for our case study are quite low but in general, the size of the search problem is all possible state instantiations (Z) of the fault detection and reconfiguration problem states. If the calls to

fault detection and reconfiguration routines are frequent for an extended mission time, then Livingstone will lose the computational advantage over the fault tolerance in CFT-SOAP. Finally, although the computational complexity of value iteration is the square of the number of states (N) times the number of actions (M), in CFT-SOAP, not all states transition to all other states and in fact there are two possible results for each action in almost all the MDPs involved hence reducing the computational complexity to two times the number of states times the number of actions. Also, the online computational complexity is a function of MDP state-space size.

Table 7-13: Comparison of computational complexity

Solution Type	CFT-SOAP	ASPEN-Livingstone
Planning	offline: ~O(2NM)) online: ~O(n)	offline: None online: ~O(QC!)
Fault Detection	offline: ~O(2NM)) online: ~O(n)	offline: None online: ~O(Z)
Reconfiguration	offline: ~O(2NM) online: ~O(n)	offline: None online: ~O(Z)

As for the robustness of the solution, the CFT-SOAP has an advantage over ASPEN-Livingstone since an MDP policy has a response for every possible state whereas the plan generated by ASPEN may only be responsive to part of the state-space, and a state outside this space may be encountered due to an anomalous event. Also, since the planning policy in CFT-SOAP considers fault probabilities, CFT-SOAP generates activities that reduce the chances of failure through explicit fault detection and reconfiguration capability. This feature is not available in the ASPEN model.

7.7 Concluding Remarks

The FUSE case study presented in this chapter provides an illustration of how to model and decompose a CFT-SOAP model for the real-world space missions such that the computation of the solution becomes tractable. Also presented in this chapter is a comparison between CFT-SOAP and alternate technologies i.e. ASPEN-Livingstone. There are three major differences in the alternate technologies. The first difference is that the planning in CFT-SOAP framework accounts for the probabilities of failures and the rewards obtained from the science data whereas the planning in ASPEN does not take into account these factors. The second major difference is that the fault tolerance in the CFT-SOAP is comprehensive i.e. accounts for the fault information obtained from both logic-based and dynamics-based fault detection whereas the formulation in Livingstone does not take into account the dynamics-based fault information. The third difference is that CFT-SOAP produces policies that include an optimal action for every possible situation whereas, in ASPEN and Livingstone, every new fault event requires online calculation of the new plan and new system configuration. Although CFT-SOAP is more computationally-complex than ASPEN or Livingstone, we propose the CFT-SOAP MDP policies be developed offline, or worst-case (MDP failure state reached) built on the ground while the spacecraft is safed. The computations for ASPEN and Livingstone can be performed online, on the spacecraft, but may require additional support from the ground as any failure state the CFT-SOAP policy doesn't cover will also not likely be covered by onboard ASPEN and Livingstone models.

Chapter 8

Conclusions and Future Directions

In the dissertation, a Markov Decision Process (MDP)-based approach to comprehensive fault tolerance and science optimal attitude planning for spacecraft applications has been presented. To reduce the computational complexity involved in solving each MDP, we demonstrated that our problem can be decomposed into multiple smaller MDPs. These MDPs included planning MDPs for different fault scenarios, a fault detection MDP for detecting faults based on a logic-based model of the spacecraft, a conflict resolution MDP for resolving the conflicts between fault information from the logic-based model and the dynamics-based model of the spacecraft, and a reconfiguration MDP that incorporates the relative importance of the mission objectives versus the safety of the spacecraft. Approximate Dynamic Programming (ADP) methods for the decomposition of the planning and fault detection MDPs have also been presented.

To illustrate the performance of the MDP-based frameworks and ADP methods, several case studies were presented. These case studies have revealed the important features of the CFT-SOAP framework and the behavior of the resulting optimal policies in response to the changes in the design parameters. A major case study based on the Far Ultraviolet Spectroscopic Explorer (FUSE) mission was presented. Our approach was also compared

with existing alternate approaches for planning and fault tolerance in the context of FUSE.

Below, specific conclusions are summarized from the work presented in the thesis. Related future research directions are also discussed.

8.1 Conclusions

8.1.1 Fault Tolerant Mission Planning

Fault-tolerant mission planning can minimize disruptions and extend the life of space missions with nominal additional effort mostly incurred during the mission design and development phase. Since a policy, generated in our formulation with an MDP, is capable of reacting to off-nominal or anomalous situations (depending upon MDP formulation used), the operational cost of space missions can be reduced. This is useful for missions where contact with the ground station is expensive, unreliable, or infrequent.

The incorporation of science reward along with costs and failure probabilities associated with mission-related actions enables the plan to be more comprehensive and robust than if manually specified as a sequence augmented by a "safing" capability that ceases mission execution in off-nominal situations. Also, since the plan includes responses under various failure scenarios, re-planning is only required when scenarios, failures, or new tasks not considered in the original planning phase are encountered. Additional capabilities of fault reconfiguration and dual fault detection by logic-based and physics-based modules ensure that the policy is executed with the best spacecraft control and compositional configuration possible given specified cost, reward, and threshold parameters.

The selection of MDP states and design parameters in reward and cost functions is a crucial step that requires understanding of the specific mission requirements, and the spacecraft used in the mission.

8.1.2 Computational Issues

A serious concern with using MDP formulation is its computational complexity which can be significant even for an off-board implementation. To reduce the computational complexity, decomposition of the MDP into smaller problems has been pursued along with the application of Approximate Dynamic Programming (ADP) algorithms. Specifically, one can make use of the structure of the mission tasks and constraints on their execution to reduce the state-space of associated MDPs. Some methods that exploit these properties were introduced in chapters 4 and 6, for example capitalizing on the separation of tasks when their observation windows do not overlap. Also by defining actions to be context-dependent, the action space is reduced significantly. Our proposed ADP algorithms have shown good performance with a significant reduction in computational complexity. With the rapidly growing computing power, the capability to generate MDP policies for large state dimensions is improving. This facilitates the implementation of the MDP-based presented approach in this thesis.

8.1.3 Implementation Issues

Although real-time execution can be more robust, potentially requiring less operational support once deployed, implementation of the presented MDP framework presented in this thesis will likely require more capable computational hardware on the spacecraft and also additional engineers in the design team to carefully build the models on which the MDPs are based. On the ground, computational resources will be required to build

policies from MDPs with large state-space sizes. On the spacecraft, policies covering large state-space sets must be executed. Communication resources must also support uploading new policies to the spacecraft as needed. The overview of the full CFT-SOAP implementation presented in Chapter 3 can be used as a guide to understand the complexity of programming and deploying the full implementation. In terms of engineering design team, additional experts will be necessary who will be responsible for analyzing the mission environment, durability and life expectancy of spacecraft components, mission costs in terms of fuel and energy given the control laws designed for various failure situations, reconfiguration options and effects of using each reconfiguration, probabilities of false alarms and missed detections for the fault detection and diagnosis algorithms used, computational complexity, and pertinent interactions within the decomposed systems that must be considered prior to building a trusted set of policies.

8.2 Future Directions

8.2.1 ADP Algorithms: Reduction of Computational Complexity

Further use of Approximate Dynamic Programming (ADP) techniques to reduce the computational complexity deserves further attention as it may lead to more efficient algorithms for implementing fault tolerant planning for real-world space missions. Researchers have made good progress in the field of Approximate Dynamic Programming, but most results are either for general use or for applications different from spacecraft, thus do not consider the challenges associated with a space mission. Developing ADP for specific spacecraft applications or mission types can ultimately contribute to a reusable (mission-independent) infrastructure to facilitiate future fault-

tolerant spacecraft deployments without the development costs that will be incurred for the first such mission. For example, the proposed ADP schemes in Chapters 4 and 5 already make use of the specific structure of the state space typically found in spacecraft mission operations to obtain a reasonable decomposition. Further matured schemes and methods could be developed that will enable comprehensive fault tolerance planning for space missions at substantially lower cost.

8.2.2 Receding Horizon Implementation and Online Learning

Another promising direction includes the development of onboard learning mechanisms for design parameters involved in the proposed algorithms. There are quite a few design parameters involved in the proposed MDPs. Finding good values for these parameters is a very difficult task; one can only find good approximations in most cases. Therefore it will be desirable to have an adaptive learning mechanism that can improve the parameters online based on observed data to enable adjustment or recalculation of optimal policies (in a receding horizon sense) based on improved parameters. If such updates occur onboard, this obviously will require more computational power that is currently available onboard spacecraft.

8.2.3 Developing MDP Frameworks for more Complex Space Missions

In this thesis, a specific mission type has been considered to convey the main ideas associated with CFT-SOAP in a clear fashion. In general, the MDP formulations can be modified for different types of space missions. For example, collaborative fault detection and comprehensive reconfiguration can be performed using the MDP formulations for missions involving interplanetary orbital maneuvers, reentry, rover deployment, etc. From the Aerospace Engineering perspective, the deterministic and predictable nature of

decisions and behaviors offered by CFT-SOAP can be an advantage for implementation. NASA's now decade-old demonstration of Remote Agent (RA) illustrated both the power and challenges of onboard deliberation.

8.2.4 Extending the Approach towards Non-Aerospace Applications

The ideas of fault tolerant planning, collaborative fault detection, conflict resolution, and comprehensive reconfiguration are extendable to other important applications especially in the automation and robotics industries. Even in newer smart cars, automated fault detection and reconfiguration can add vital value. Industrial and commercial robots, unmanned aircraft systems (UAS), and even toys can be developed with built-in optimal responses for a number of situations (states).

CFT-SOAP can be generalized for other types of missions with appropriate changes in the formulation. The problem of validation and verification (V&V) becomes ever more difficult as model and architecture complexity increases. Even when software implementing algorithms internal to each layer completes a V&V process, the new communications, including arbitration between potentially-disparate conclusions, introduces a new challenge. From a practical implementation standpoint, while the integration of deliberative compositional and dynamics-based algorithms enables capture and management of a more comprehensive fault set, increased complexity also increases risk of unanticipated execution sequences due to unpredicted interactions. A formidable but surmountable challenge is then to ensure the integrated system is validated.

As a multidisciplinary architecture, CFT-SOAP will require team-based adaptation to any domain, with composition of experts in both symbolic inference and adaptation (AI)

and physics-based control systems. This may be a negative for small projects, although the author argues that any goal-based system with nontrivial dynamics requires participation from both communities today, just in segregated modules presumed to work with little to no knowledge of the other.

References

[1] "Electromagnetic Storm Hits Intelsat Satellite", *Space News*, Vol. 5, No 5, January 31-February 6, 1994, p. 3.

[2] "Spacecraft Anomaly Database", Version. ANOM5I, National Geophysical Data Center, Solar-Terrestrial Physics Division, Boulder CO, March 1994.

[3] Aberkane, S. Ponsart, J.C. Rodrigues, M. and Sauter, D. "Output feedback control of a class of stochastic hybrid systems", *Automatica*, Elsevier 2008.

[4] 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.

[5] Adams, L., "Proton Induced Upsets in the Low Altitude Polar Orbit," *IEEE Transactions on Nuclear Science*, Vol. 36, No. 6, December 1989, pp. 2339-2343.

[6] Antsaklis, P.J. "A Brief introduction to the Theory and Applications of Hybrid Systems". *Proceedings of the IEEE, Special Issue on Hybrid Systems: Theory and Applications,* Vol. 88, No. 7, pp. 879-887, July 2000.

[7] Atlas, A. K., and Bestavros, A., "Statistical Rate Monotonic Scheduling", *Proceedings of the 19th IEEE Real-Time Systems Symposium*, 1998, pp. 123-132.

[8] Bate, Roger et al, *Fundamentals of Astrodynamics*, Dover Publications, INC. New York copyright 1971.

[9] Bedingfield, K.L., Leach, R. D., and Alexander, M. B., "Spacecraft System Failures and Anomalies Attributed to the Natural Space Environment". *National Aeronautics and Space Administration*: Marshall Space Flight Center, Alabama, 1996.

[10] Bertsekas, D. and Tsitsiklis, J., *Neuro-dynamic programming*, Athena Scientific, Belmont, MA, 1996.

[11] Bialke, B., and Dorsey, G., "FUSE Reaction Wheel Torque Anomaly Resolution," *Advances in the Astronautical Sciences* 107, pp. 441-458, 2001.

[12] Blanke, M., Izadi-Zamanabadi, R., & Lootsma, T. F., "Fault monitoring and re-configurable control for a ship propulsion plant", *International Journal of Adaptive Control and Signal Processing*, 12(8), 671–688.

[13] Blanke, M., Izadi-Zamanabadi, R., Bogh, R., & Lunau, Z. P., "Fault Tolerant control systems—A holistic view", *Control Engineering Practice*, 5(5), 693–702, 1997.

[14] Boutilier, C. Brafman, R.I., and Geib, C. Prioritized goal decomposition of Markov decision processes: Toward a synthesis of classical and decision theoretic planning, in: Proc. IJCAI-97, Nagoya, Japan, 1997, pp. 1162–1165.

[15] Boutilier, C. Dean, T. Hanks, S. Decision-Theoretic Planning: Structural Assumptions and Computational Leverage *Journal of Artificial Intelligence Research 11 (1999)* 1-94.

[16] Boutilier, C. et al., "Prioritized Goal Decomposition of Markov Decision Processes: Toward a Synthesis of Classical and Decision Theoretic Planning", *Proc. 15th Intl. Joint Conf. on AI (IJCAI-97),* Nagoya, August, 1997.

[17] Brailsford, S.C. et al. "Constraint Satisfaction Problems: Algorithms and Applications". *European Journal of Operational Research* 119 (1999) 557-581.

[18] Burlton, Bruce, "The Rescue of Anik E2", *Canadian Aeronautics and Space Journal*, Vol. 41, Ottawa : CASI, 1995.

[19] Castet, J. F. and Saleh, J. H., "Satellite and satellite subsystems reliability: Statistical data analysis and modeling", *Reliability Engineering and System Safety*, volume 94, pages 1718-1728, 2009

[20] Castet, J. F. and Saleh, J. H., "Single versus mixture Weibull distributions for nonparametric satellite reliability", *Reliability Engineering and System Safety*, volume 95, pages 295-300, 2010

[21] Chamseddine, Abbas, Noura, Hassan and Ouladsine, M. "Sensor Fault Detection, Identification and Fault Tolerant Control: Application to Active Suspension". 1-4244-0210-7/06 2006 IEEE.

[22] Chien, S. Knight, R. Stechert, A. Sherwood, R. Rabideau, G. "Using Iterative Repair to Increase the Responsiveness of Planning and Scheduling for Autonomous Spacecraft". *International Joint Conference on Artificial Intelligence (IJCAI 1999)*. Stockholm, Sweden. August 1999.

[23] Clark, R.N., Fosth, D.C. and Walton, W.M., "Detecting Instrument Malfunctions in Control Systems," *IEEE Trans. Aerospace and Electronic Systems*, IEEE, Vol. AES-11, 465-473, 1975.

[24] Conway, R. W. et al. *Theory of Scheduling*, Dover Publications Inc., 31 East 2nd Street, Mineola, N.Y. 11501, 2003.

[25] Crassidis, J.L. and Markley, F.L. "A Minimum Model Error Approach for Attitude Estimation". *AIAA Journal of Guidance Control and Dynamics*, 20 (6) (1997) 1241-1247.

[26] Crassidis, J.L. and Markley, F.L. "Attitude Estimation Using Modified Rodrigues Parameters" Proceedings of the Flight Mechanics/Estimation Theory Symposium, (NASA/CP-1996-3333)NASA-Goddard Space Flight Center, Greenbelt, MD, 1996, pp. 71–83.

[27] Crassidis, J.L. and Markley, F.L. "Unscented Filtering for Spacecraft Attitude Estimation". AIAA Journal of Guidance Control and Dynamics, 20 (4) (2003) 536-542.

[28] Crassidis, J.L. et al."A Survey of Nonlinear Attitude Estimation Methods". AIAA Journal of Guidance, Control, and Dynamics, 30 (1): 12-28, January 2007.

[29] Das, S. et al. "Reconfigurable Magnetic Attitude Control of Earth Pointing Satellites". Proceedings of Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering, 2010 224:1309. Published by SAGE. DOI: 10.1243/09544100JAERO681.

[30] Dean, T. and Givan, R. “Model Minimization in Markov Decision Processes”. AAAI-97 Proceedings. Copyright © 1997, pp 106-111.

[31] Dean, T. and Kanazawa, K., “A model for Reasoning about Persistence and Causation”, *Computational Intelligence*, *5.* 142-150 (1989).

[32] Dean, T.L., McDermott, D.V. “Temporal data base management”, *Artificial Intelligence* 32 (1987) l-55.

[33] Dechter, R. “Temporal Constraint Networks”. *Artificial Intelligence*, 49 (1991) 61-95. Elsevier Science Publishers B.V.

[34] Dechter, R. and Pearl, J. “Network Based Heuristics for Constraint Satisfaction Problems”. *Artificial Intelligence*, 34 (1) (1987),pp 1 – 38.

[35] Dechter, R. *Constraint Processing*, Morgan Kaufmann Publishers an Imprint of Elsevier Science, San Francisco, California. Copyright 2003 by Elsevier Science (USA).

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

[37] Ding, S. and Li, S. “Stabilization of the Attitude of a Rigid Spacecraft with External Disturbances using Finite-time Control Techniques”. Aerospace Science and Technology, 13 (2009) 256-265. Copyright by Elsevier Masson SAS 2009.

[38] Elsen, William, G., *Orbital Anomalies in Goddard Spacecraft for CY 1989*, Assurance Requirements Office, Office of Flight Assurance, NASA Goddard Space Flight Center, July 1990.

[39] Frank, P. M., "Fault diagnosis in dynamic systems using analytical and knowledge-based redundancy—a survey and some new results", *Automatica*, 26(3), 459–474, 1990.

[40] Frank, P.M. and Ding, X., "Survey of Robust Residual Generating and Evaluation Methods in Observer-Based Fault Detection Systems," *Journal of Process Control*, Elsevier Ltd, Vol. 37, No. 6, 403-424, 1997.

[41] Garret, H., and Whittlesey, "Environment Induced Anomalies on the TDRSS and the Role of Spacecraft Charging," 28th Aerospace Sciences Meeting, January 8-11, Reno, Nevada.

[42] Garret, Henry, Berry, "The Charging Of Spacecraft Surfaces," *Reviews of Geophysics and Space Physics*, Vol. 19, No. 4, November, 1981, p. 577-616.

[43] Gat, E., "ESL: a language for supporting robust plan execution in embedded autonomous agents", in: L. Pryor (Ed.), Proc. AAAI Fall Symposium on Plan Execution, AAAI Press, 1996.

[44] Goldman, C. V. and Zilberstein, S., "Communication-Based Decomposition Mechanisms for Decentralized MDPs", *Journal of Artificial Intelligence Research*, 32 (2008) 169-202.

[45] Haralick, R.M. and Elliott, G.L. "Increasing Tree Search Efficiency for Constraint Satisfaction Problems". Artificial Intelligence 14 (1980) 263-313, Copyright © by North-Holland Publishing Company.

[46] Havelund, K., Lowry, M. et al "Formal Analysis of the Remote Agent Before and After Flight." *In Proceedings of the 5th NASA Langley Formal Methods Workshop*, June 2000.

[47] Hecht, H. and Hecht, M., "Reliability Prediction for Spacecraft", *Rome Air Development Center, Griffiss Air Force Base*, Final technical report RADC-TR-85-229, New York, December 1985.

[48] Ho, L.-W., & Yen, G. G. (2002). "Reconfigurable control system design for fault diagnosis and accommodation", *International Journal of Neural Systems*, 12(6), 497–520.

[49] Ho, Y. C. and Chu, K. C. "Team decision theory and information structures in optimal control problems-part I", *IEEE Transactions on Automatic Control,* 17 (1972), pp. 15 - 22.

[50] Hughes, David, "Telsat Succeeds in Anik E2 Rescue," Aviation Week & Space Technology, July 4, 1994, p. 32.

[51] Hughes, Peter "Spacecraft Attitude Dynamics" Dover Publications, INC. New York copyright 1986, 2004 Peter Hughes

[52] Johnson, M. D. and Miller, G. E., "Spike: Intelligent Scheduling of Hubble Space Telescope Observations", edited by M. Zweben and M. Fox, *Intelligent Scheduling*, Morgan Kaufmann Publishers, San Francisco, CA, 1994.

[53] Jonsson, A., & Barto, A., "A causal approach to hierarchical decomposition of factored MDPs", *Proceedings of the Twenty Second International Conference on Machine Learning* (ICML 05), 2005.

[54] Knapp, Bill, "Telsat Ponders Using Thrusters To Salvage Anik," Space News, Vol. 5, No. 5, January 31- February 6, 1994, p. 1.

[55] Krishnan, H. McClamroch, N.H. and Reyhanoglu, M. "Attutude Stabilization of a Rigid Spacecraft Using Two Control Torques: A Nonlinear Control

Approach Based on the Spacecraft Attitude Dynamics". Automatica, Vol. 30, No. 12, December, 1994, 1885-1897.

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

[57] Kruk, L. et al. "Earliest-Deadline-First Service In Heavy-Traffic", *The annuals of applied probability*, 2004, Vol 14, No. 3, pp. 1306-1352.

[58] Kumar, P. R. and Varaiya, P., *Stochastic Systems: Estimation, Identification, and Adaptive Control*, Prentice Hall Inc., Englewood Cliffs, New Jersey 07632, 1986.

[59] Kurk, J. W. et al, "FUSE Fine Error Sensor Optical Performance", *Proc. SPIE* 4139, 2000.

[60] Labinaz, Gino et al. "A Survey of Modeling and Control of Hybrid Systems". *International Fedration of Automatic Control 1997*, published in Great Britan, S0066-4138(97)00019-0, Vol 21, pp. 79-92.

[61] Leffers, E.J. et al "Kalman Filtering for Spacecraft Attitude Estimation" AIAA 20th Aerospace Science Meeting, Orlando, Florida, January 11-14, 1982.

[62] Lovera, M. and Astolfi, A. "Spacecraft Attitude Control using Magnetic Actuators". Automatica, 40 (2004) 1405-1414. Copyright 2004 Elsevier Ltd.

[63] Marschak, J. "Elements for a theory of teams". Management Sci., 1 (1955), pp. 127 - 137.

[64] 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.

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

[66] Muscettola, N. "HSTS: Integrating planning and scheduling". In Mark Fox and Monte Zweben, editors, *Intelligent Scheduling*. Morgan Kaufmann, 1994.

[67] Muscettola, N. Smith, B. Chien, C. Fry, C. Rabideau, G. Rajan, K. Yan, D. "On-board planning for autonomous spacecraft", in: *Proc. 4th International Symposium on Artificial Intelligence, Robotics. And Automation for Space* (i-SAIRAS), Tokyo, Japan, August 1997.

[68] Muscettola, N., Nayak, P., et al "Remote Agent: To Boldly Go Where No AI System Has Gone Before". Artificial Intelligence, 103(1-2):5--48, 1998.

[69] Nasir, A. Atkins, E.M. Kolmanovsky, I.V. "Science-optimal Spacecraft Attitude Maneuvering While Accounting for the Failure Mode". 18th IFAC World Congress. Milano, Italy. August 29th to September 2nd 2011.

[70] 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).

[71] Nasir, A., Atkins, E.M., and Kolmanovsky, I.V. "Conflict Resolution Algorithms for Fault Detection and Diagnosis" *AIAA Infotech@Aerospace Conference*, St. Louis, Missouri, March. 29-31, 2011 (AIAA-2011-1587).

[72] Park, B.G. and Kwon, W.H., "Robust one-step receding horizon control of discrete-time Markovian jump uncertain systems", *Automatica*, 38 (2002) 1229-1235.

[73] Patton R. J., Lopez-Toribio C. J., & Uppal F. J., "Artificial Intelligence Approaches to Fault Diagnosis," *Applied Mathematics and Computer Science*, Technical University of Zielona Gora, Poland, Vol. 9, No. 3, 471-518, 1999.

[74] Patton, R. J., "Fault-tolerant control: The 1997 situation", *In Proceedings of the 3rd IFAC symposium on fault detection, supervision and safety for technical processes*, (pp. 1033–1055), August, 1997.

[75] Patton, R. J., Frank, P. M., & Clark, R. N., *Fault diagnosis in dynamic systems: Theory and applications*. Englewood Cliffs, NJ: Prentice-Hall, 1989.

[76] Pell, B. et al "A Hybrid Procedural/Deductive Executive for Autonomous Spacecraft". *Autonomous Agents and Multi Agent Systems*, 2, 7-22(1999) Kluwer Academic Publishers, 1999.

[77] Pinedo, M. L. *Scheduling: Theory, Algorithms, and Systems*, 4th edition, Springer Science+Business Media, LLC, 233, Spring Street, New York, NY, 10013, USA, 2012.

[78] Potocnik, B. et al. "Model Predictive Control of Discrete-time Hybrid Systems with Discrete Inputs". *ISA Transactions*, 44 (2005), 199-211.

[79] Powell, W. B., *Approximate Dynamic Programming: Solving the Curses of Dimensionality*. New York: Wiley, 2007.

[80] Powell, W.B. "What You Should Know About Approximate Dynamic Programming". *Naval Research Logistics*, Vol. 56 (2009) pp 239-249,

published online 24 February 2009 in *Wiley InterScience*, DOI 10.1002/nav.20347.

[81] Pritchard, B. E., Swift, G. M., and Johnston, A. H., "Radiation effects, predicted, observed and compared for spacecraft systems," *IEEE NSREC 2002 Data Workshop Proc*., 2002, pp. 7–17.

[82] Pulecci, T. et al. "Classical vs Modern Magnetic Attitude Control Design: A Case Study". GNC 2008 7th International ESA Conference, Tralee, County Kerry, Ireland, 2008.

[83] Purvis, C.K. et al., "Design Guidelines for Assessing and Controlling Spacecraft Charging Effects". *National Aeronautics and Space Administration*, Technical Paper 2361, 1984.

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

[85] Rabideau, G. Knight, R. Chien, S. Fukunaga, A. Govindjee, A. "Iterative Repair Planning for Spacecraft Operations in the ASPEN System". *International Symposium on Artificial Intelligence Robotics and Automation in Space (ISAIRAS 1999)*, Noordwijk, The Netherlands. June 1999.

[86] Radner, R. "Team decision problems". *Ann. Math. Statist.*, 33 (1962), pp. 857 - 881.

[87] 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.

[88] Roberts, Bryce A., et al, "Three-axis Attitude Control with Two Reaction Wheels and Magnetic Torquer Bars," *AIAA Guidance, Navigation, and Control Conference and Exhibit*, Paper 5245, Providence, Rhode Island, 16-19 August 2004.

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

[90] Sahnow, D. J. et al, "On-orbit Performance of the Far Ultraviolet Spectroscopic Explorer Satellite" *The Astrophysical Journal*, 538: L7-L11, July, 2000.

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

[92] Shima, T. and Rasmussen, S. *UAV Cooperative Decision and Control Challenges and Practical Approaches*, Copyright © 2009 by Society of Industrial and Applied Mathematics.

[93] Shuster, M.D. "A Survey of Attitude Representations". *Journal of Astronautical Sciences*, Vol 41, No 4, October-December, 1993, pp 439-517.

[94] Silani, E. and Lovera, M. "Magnetic Spacecraft Attitude Control: A Survey and Some New Results". *Control Engineering Practice*, 13 (2005) 357-371. Copyright Elsevier Ltd.

[95] Smith, B., Feather, Martin S. and Muscettola, N. "Challenges and Methods in Testing the Remote Agent Planner", *Proceedings of the Fifth International Conference on Artificial Intelligence Planning Systems*, Breckenridge, CO.

[96] Sullivan, B. R. and Akin, D. L., " A Aurvey of Serviceable Spacecraft Failures", *AIAA* 2001-4540

[97] Sutton, R. and Barto, A., *Reinforcement learning*, The MIT Press, Cambridge, Massachusetts, 1998.

[98] Wadham, P., N., "The Effects of Electrostatic Discharge Phenomena on Telsat's Domestic Communications Satellites," *AGARD*, The Aerospace Environment at Altitude and Its Implications for Spacecraft Charging, 1987, p. 21-18.

[99] Walker, B.K. and Gai, E., "Fault Detection Threshold Determination Technique Using Markov Theory," *Journal of Guidance, Control and Dynamics*, AIAA, Vol. 2, 313-319, July-Aug. 1979.

[100] Web link: http://www.weibull.com/hotwire/issue13/relbasics13.htm

[101] Wertz, J.R. and Larson, W. J., *Space Mission Analysis and Design*. El Segundo, California: Microcosm Press, 2003.

[102] Wie, Bong "Space Vehicle Dynamics and Control" AIAA Education Series 1998.

[103] Wilkinson, D., "TDRS-1 Single Event Upsets and the Effect of the Space Environment," *IEEE Transactions on Nuclear Science*, Vol. 38, No. 6, December 1991.

[104] 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.

[105] Witsenhausen, H. "Equivalent stochastic control problems". *Mathematics of Control, Signals, and Systems* (MCSS), 1 (1988), pp. 3-11.

[106] Xiong, M., Wang, Q., and Ramamritham, K., "On earliest deadline first scheduling for temporal consistency maintenance," *Real-Time Systems*, 2008.

[107] Xu, R. et al. "Multi-Agent Planning System for Spacecraft". *Proceedings of the Second International Conference on Machine Learning and Cybernetics*, Xi'an, 2-5 November 2003, pp 1995-1999.

[108] Yoshikawa, T. "Decomposition of dynamic team decision problems". *Automatic Control, IEEE Transactions on*, 23 (1978), pp. 627 - 632.

[109] Zalewski, J., "What Every Engineer Needs to Know about Rate Monotonic Scheduling: A Tutotial", *IEEE magazine-95/1*, IEEE Computer Society Press, 1995.

[110] Zentgraf, P and Reggio, D. "Magnetic Rate Damping for Satellites in LEO". *32nd Annual AAS Guidance and Control Conference*, Jan 31 – Feb 4, 2009, Breckenridge, Colorado.

[111] 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.

[112] 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.

[113] Zhang, Y.M. and Jiang, J. “Fault Tolerant Control System Design with Explicit Consideration of Performance Degradation”, *IEEE Transactions on Aerospace and Elctronic Systems*, 2003, 37 (3).

[114] Zhou, D.H. and Frank, P.M. “Fault Diagnostics and Fault Tolerant Control”. *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 34, No. 2, April 1998.

[115] Zweben, M. and Mark S., *Fox Intelligent Scheduling* (Chapter 6), Morgan Kaufman Publishers, San Francisco, California, 1994.