# ACCURATE BELIEF STATE UPDATE FOR PROBABILISTIC CONSTRAINT AUTOMATA

by

Oliver Borelli Martin

B.S., Department of Aerospace Engineering
California Polytechnic State University, 2002

SUBMITTED TO THE DEPARTMENT OF AERONAUTICS AND ASTRONAUTICS
IN PARTIAL FULFILLMENT OF THE REQUIREMENT FOR THE DEGREE OF

MASTER OF SCIENCE

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2005

Signature of Author . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
Department of Aeronautics and Astronautics
May 20, 2005

Certified by . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
Brian C. Williams
Associate Professor of Aeronautics and Astronautics
Thesis Supervisor

Accepted by . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
Jaime Peraire
Professor of Aeronautics and Astronautics
Chair, Committee on Graduate Students

# Accurate Belief State Update for Probabilistic Constraint Automata

by

Oliver Borelli Martin

Submitted to the Department of Aeronautics and Astronautics
on May 20, 2005, in partial fulfillment of the
requirements for the degree of
Master of Science at the
Massachusetts Institute of Technology

## Abstract

As autonomous spacecraft and other robotic systems grow increasingly complex, there is a pressing need for capabilities that more accurately monitor and diagnose system state while maintaining reactivity. Mode estimation addresses this problem by reasoning over declarative models of the physical plant, represented as a factored variant of Hidden Markov Models (HMMs), called *Probabilistic Concurrent Constraint Automata (PCCA)*. Previous mode estimation approaches track a set of most likely PCCA state trajectories, enumerating them in order of trajectory probability. Although *Best-First Trajectory Enumeration (BFTE)* is efficient, ignoring the additional trajectories that lead to the same target state can significantly underestimate the true state probability and result in misdiagnosis. This thesis introduces two innovative belief state approximation techniques, called *Best-First Belief State Enumeration (BFBSE)* and *Best-First Belief State Update (BFBSU)*, that address this limitation by computing estimate probabilities *directly* from the HMM belief state update equations. Theoretical and empirical results show that BFBSE and BFBSU significantly increase estimator accuracy, uses less memory, and has no increase in computation time when enumerating a moderate number of estimates for the approximate belief state of subsystem sized models.

Thesis Supervisor: Brian C. Williams
Title: Associate Professor of Aeronautics and Astronautics

# Acknowledgments

This thesis would not have been possible without the persistence and patience of Brian Williams, Michel Ingham, and Seung Chung. It has been a privilege to work with you and I appreciate the continuous guidance and encouragement that you have provided me.

To all my family and friends: Please forgive me for my negligence over the past few years. I look forward to spending much more time with everyone in the future.

This research was supported in part by NASA's Cross Enterprise Technology Development program under contract NAG2-1466, and by the Jet Propulsion Laboratory, through the JPL Director's Research and Development Fund under contract 1261107.

# Contents

**1 Introduction** **13**

1.1 Previous Work . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 13

1.2 Thesis Outline . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 14

**2 Decent Stage IMU** **17**

2.1 Simple IMU Example . . . . . . . . . . . . . . . . . . . . . . . . . . 18

2.1.1 The IMU Component Model . . . . . . . . . . . . . . . . . . . . 19

2.1.2 The Power Switch Component Model . . . . . . . . . . . . . . . 21

2.1.3 The Timer Component Model . . . . . . . . . . . . . . . . . . . 21

2.2 PCCA Plant Model . . . . . . . . . . . . . . . . . . . . . . . . . . . 22

2.2.1 PCCA Formalism . . . . . . . . . . . . . . . . . . . . . . . . . . 23

2.2.2 Example: The IMU System PCCA Plant Model . . . . . . . . . . 24

**3 Estimation of PCCA** **29**

3.1 Belief State Update . . . . . . . . . . . . . . . . . . . . . . . . . . . 29

3.1.1 PCCA Transition and Observation Probabilities . . . . . . . . . . 30

3.1.2 Example: IMU System Belief State Update . . . . . . . . . . . . 32

3.2 Optimal Constraint Satisfaction Problems . . . . . . . . . . . . . . 35

3.2.1 Example: IMU System OCSP Belief State Update . . . . . . . . 37

**4 Approximate Estimation of PCCA** **39**

4.1 Belief State Update Approximations . . . . . . . . . . . . . . . . . 39

4.1.1 Exponential Belief State Approximation . . . . . . . . . . . . . . 40

4.1.2 Trajectory Approximation . . . . . . . . . . . . . . . . . . . . . 41

4.1.3 Observation Approximation . . . . . 43

4.2 Best-First Solutions using OPSAT . . . . . 45

4.2.1 *Conflict-directed A** . . . . . 45

4.2.2 Estimation using OPSAT . . . . . 47

4.2.3 Example: Most Likely Trajectory for the IMU System . . . . . 48

4.3 *Best-First Belief State Enumeration* . . . . . 50

4.3.1 BFBSE Heuristic Function . . . . . 50

4.3.2 Example: BFBSE for the IMU Plant . . . . . 52

4.4 *Best-First Belief State Update* . . . . . 56

4.4.1 Observation Probability Rules . . . . . 58

4.4.2 Offline Generation of Observation Probability Rules . . . . . 60

4.4.3 Online BFBSU using Observation Probability Rules . . . . . 64

4.4.4 Example: BFBSU for the IMU Plant . . . . . 66

**5 Results and Discussion 71**

5.1 Space and Time Complexity . . . . . 71

5.2 Experimental Results . . . . . 73

5.2.1 Accuracy . . . . . 74

5.2.2 Performance . . . . . 77

**6 Conclusion 81**

6.1 Summary of Results . . . . . 82

6.2 Future Work . . . . . 82

**A IMU System Constraint Automata Definitions 85**

A.1 The Power Switch Constraint Automaton, $\mathcal{A}_{ps}$ . . . . . 85

A.2 The Timer Constraint Automaton, $\mathcal{A}_t$ . . . . . 87

A.3 Compiled IMU Plant Model Dissents . . . . . 88

**B *Best-First Trajectory Enumeration* 89**

B.1 BFTE Heuristic Function . . . . . 90

# List of Figures

2-1 Mars Science Laboratory sky-crane EDL . . . . . . . . . . . . . . . . . . 18

2-2 Simple IMU system block diagram. . . . . . . . . . . . . . . . . . . . . . . 19

2-3 Model of IMU operational modes. . . . . . . . . . . . . . . . . . . . . . . 20

2-4 Model of Power Switch operational modes. . . . . . . . . . . . . . . . . . 21

2-5 Model of Timer operational modes. . . . . . . . . . . . . . . . . . . . . . 22

2-6 IMU constraint automaton, $\mathcal{A}_{imu}$. . . . . . . . . . . . . . . . . . . . . . 25

3-1 The *Trellis diagram* of the possible state evolutions over time. . . . . . . 30

3-2 Single-step exact belief state update of the IMU Plant. . . . . . . . . . . 32

3-3 Single-step OCSP exact belief state update for the IMU Plant. . . . . . . 37

4-1 Belief state probability distribution for a simple IMU Plant scenario. . . 41

4-2 Decomposing the belief state evolution into a branching structure. . . . . 41

4-3 Approximate trajectory propagation for the IMU Plant. . . . . . . . . . . 42

4-4 Approximate observation probability update for the IMU Plant. . . . . . 44

4-5 Single Trajectory Enumeration for the IMU Plant. . . . . . . . . . . . . . 49

4-6 *Best-First Belief State Enumeration* for the IMU Plant. . . . . . . . . . . 52

4-7 Iteration 1 BFBSE expansion for the IMU Plant. . . . . . . . . . . . . . 53

4-8 Iteration 2 BFBSE expansion for the IMU Plant. . . . . . . . . . . . . . 54

4-9 Iteration 3 BFBSE expansion for the IMU Plant. . . . . . . . . . . . . . 55

4-10 Iteration 4 BFBSE expansion for the IMU Plant. . . . . . . . . . . . . . 55

4-11 OPR dependency hypergraph for the IMU Plant. . . . . . . . . . . . . . 62

4-12 *Best-First Belief State Update* for the IMU Plant. . . . . . . . . . . . . . 66

4-13 Iteration 1 BFBSU expansion for the IMU Plant. . . . . . . . . . . . . . 67

4-14 Iteration 2 BFBSU expansion for the IMU Plant. . . . . . . . . . . . . . 68

4-15 Iteration 3 BFBSU expansion for the IMU Plant. . . . . . . . . . . . . . . 68

5-1 Probability density maintained over time for EO-1 model. . . . . . . . . . . 75
5-2 Probability density maintained over time for the Mars EDL model. . . . 76
5-3 Probability density maintained over time for the ST7-A model. . . . . . . 76
5-4 Single state estimate probability over time for the ST7-A model. . . . . . 77
5-5 Best-case and average-case runtime performance for the EO-1 model. . . 78
5-6 Best case and average case runtime performance for the Mars EDL model. 79
5-7 Best case and average case runtime performance for the ST7-A model. . . 79

A-1 PS constraint automaton, $\mathcal{A}_{ps}$. . . . . . . . . . . . . . . . . . . . . . . . . . 86
A-2 T constraint automaton, $\mathcal{A}_t$. . . . . . . . . . . . . . . . . . . . . . . . . . . 87

# List of Tables

2.1 $\mathcal{A}_{imu}$ Modal Constraints . . . . . . . . . . . . . . . . . . . . . . . . . 25
2.2 $\mathcal{A}_{imu}$ Transition Functions / Probabilities . . . . . . . . . . . . . . . . . . 26

3.1 Enabled Component Transitions for the IMU Plant, $\mathcal{P}$ . . . . . . . . . . 33

4.1 Results for *Best-First Trajectory Enumeration* . . . . . . . . . . . . . . . 43
4.2 Results for Approximate Observation Probability Update . . . . . . . . . 45
4.3 Enabled Component Transitions for the IMU Plant, $\mathcal{P}$ . . . . . . . . . . 53

5.1 Space and Time Complexity for BFTE, BFBSE, and BFBSU . . . . . . . 72
5.2 Experimental Model Properties . . . . . . . . . . . . . . . . . . . . . . . . 74
5.3 Number of Observation Probability Rules for each Model . . . . . . . . . 80

A.1 $\mathcal{A}_{ps}$ Modal Constraints . . . . . . . . . . . . . . . . . . . . . . . . . . . 86
A.2 $\mathcal{A}_{ps}$ Transition Functions / Probabilities . . . . . . . . . . . . . . . . . . 86
A.3 $\mathcal{A}_{t}$ Modal Constraints . . . . . . . . . . . . . . . . . . . . . . . . . . . . 87
A.4 $\mathcal{A}_{t}$ Transition Functions / Probabilities . . . . . . . . . . . . . . . . . . . 88

# Chapter 1

# Introduction

The purpose of estimation is to determine the current state of the system. An estimator infers the current state by reasoning over a model of the system dynamics along with the commands that have been executed and the resulting sensory observations. In many embedded systems, this knowledge of the current state is then used by a controller to drive the system state towards a specific target or goal. The ability for a system to accurately and reliably deduce its current state can dictate whether it is able to achieve its objectives. This is particularly important for highly complex robotic space exploration systems that operate in uncertain environments. Furthermore, deep space communication delays and severely constrained on-board computing capabilities present tremendous challenges to traditional methods of estimation in support of robust autonomous spacecraft operations.

## 1.1 Previous Work

Previous work in model-based monitoring and fault diagnosis, including GDE/Sherlock [6, 7], GDE+ [19], Livingstone [21, 15], diagnosis using model-checking [5], and Titan Mode Estimation [20], have made significant advances towards meeting these challenging performance requirements. All of these capabilities achieved reactivity, while maintaining reliability, by framing mode estimation as a best-first shortest-path problem, which can be efficiently solved using a variant of the Viterbi algorithm [10]. This approach is known as *Best-First Trajectory Enumeration (BFTE)* and works quite well when trying to determine the "most likely explanation" to a sequence of observations. Livingstone was successfully flight validated on the NASA Deep Space One probe as part of the Remote Agent Experiment in 1999 [17]. Unfortunately, approximating the current state by the most likely trajectory can significantly underestimate the true state probability and result in misdiagnosis. In addition, failure to update the estimates with valid observation probabilities places a probabilistic bias on failure modes, which can drive the estimator towards incorrect fault diagnoses during continuous nominal operations.

This thesis introduces two novel mode estimation techniques, called *Best-First Belief State Enumeration (BFBSE)* and *Best-First Belief State Update (BFBSU)*, that approximate the belief state by generating the set of most likely estimates and achieve greater accuracy than BFTE by computing the estimate probabilities *directly* from the Hidden Markov Model (HMM) belief state update equations, instead of approximating them by their trajectory probability. This contribution significantly increases the accuracy of the estimator while using less memory and less computational time; providing an enabling technology for increasingly complex space missions of the future.

## 1.2 Thesis Outline

This thesis first provides a motivating example of a small IMU system, similar to the one that will fly on the decent stage of the Mars Science Laboratory [12] in 2009. Our Probabilistic Concurrent Constraint Automata (PCCA) formalism [21, 20] is then presented in detail and the IMU Plant model is formally described. Chapter 3 reviews the exact solution to the PCCA estimation problem using the HMM belief state update equations, eluding to some practical limitations due to PCCA state space explosion. The PCCA estimation problem is then framed as an Optimal Constraint Satisfaction Problem (OCSP), providing the framework for efficient best-first enumeration of state estimates. Due to state space explosion, Chapter 4 discusses the three significant approximations that were employed in BFTE to achieve the strict computational requirements of severely constrained embedded systems, while maintaining estimate accuracy. BFBSE and BFBSU are then introduced as superior mode estimation techniques that significantly improve estimate accuracy through direct use of the HMM belief state update equations. In addition, BFBSE and BFBSU improve estimator performance by framing the PCCA estimation problem as a single OCSP, and using the observation probabilities in the search heuristic to quickly identify likely solutions and avoid sub-optimal candidates. In Chapter 5, we support our claims of improved estimator accuracy and performance through theoretical and empirical comparisons between BFTE, BFBSE, and BFBSU. Experimental data gathered from three different spacecraft subsystem models show good alignment with our theoretical expectations. In conclusion, Chapter 6 summarizes the technical contributions of this thesis and provides insight into future areas of research in the field of model-based monitoring and fault diagnosis.

# Chapter 2

# Decent Stage IMU

The Mars Science Laboratory (MSL) is NASA JPL's next generation Mars rover, which is currently slated to be launched in December of 2009. The primary science objective of MSL is to conduct in-situ analysis of Martian soil in search for organic compounds that are necessary to support life [12]. MSL is twice as long and three times as massive (3000 kg) as the Mars Exploration Rovers (MER), carrying an unprecedented 10 science instruments. Due to this large size and mass, the current airbag landing system is no longer sufficient and a novel "sky-crane" approach (shown in Figure 2-1) will be used to safely place MSL on the Martian surface. In addition, MSL will be the first Mars Lander to use precision guidance during entry, decent, and landing (EDL), in order to accurately control MSL to within a 10km x 5km, $3\sigma$ landing target error ellipse [12]. This innovative EDL sequence and precise landing target requirement places a substantial demand on the MSL EDL system, which must operate autonomously due to the 4 to 21 minute time-delay between Earth and Mars.

The Inertial Measurement Unit (IMU) is a critical component for supporting MSL EDL. An IMU is typically composed of 3 accelerometers and 3 gyroscopes, one for each axis, and is responsible for providing body-frame position and attitude measurements with a fast update rate, as necessary for real-time control. MSL will rely on an IMU attached to its decent stage (Figure 2-1) in order to make the necessary adjustments to its flight-path during entry for a safe and precise rover landing. With less than 6 minutes in the entire EDL sequence, quick and accurate monitoring and diagnosis of the IMU operational mode is essential for the success of the mission. Failure to autonomously diagnose and recover from an IMU failure mode would certainly lead to unreliable position

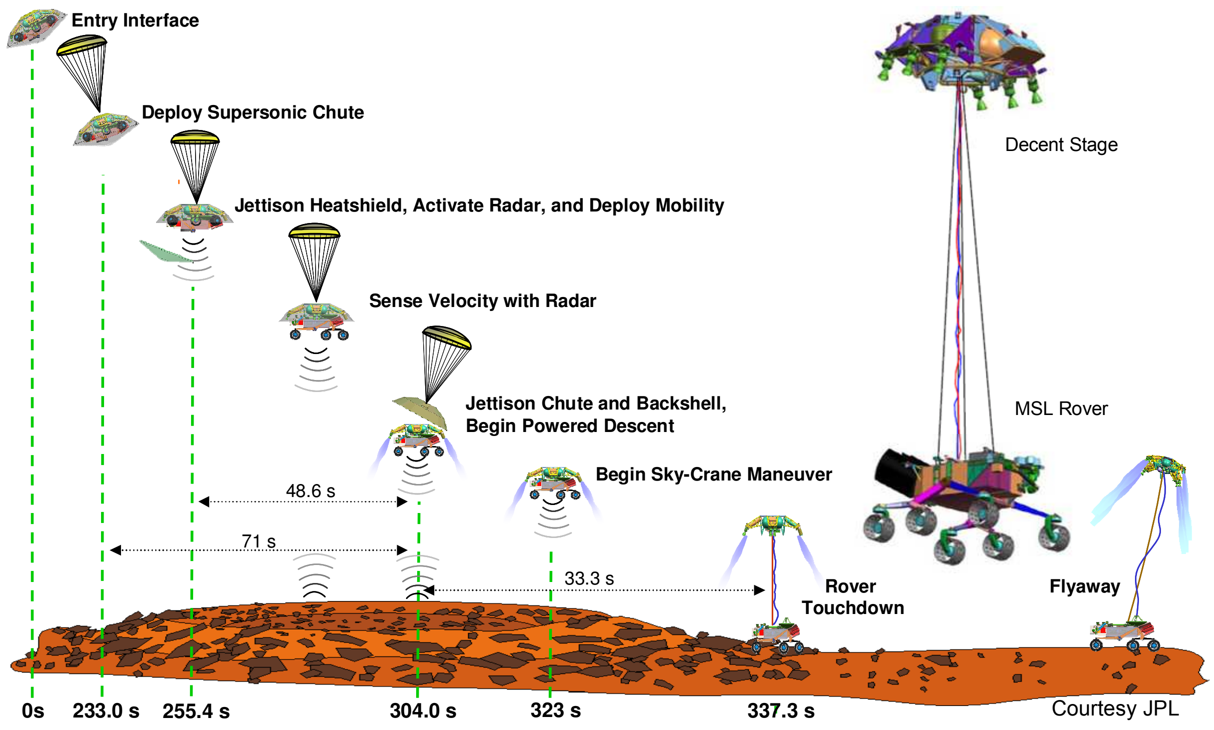


Figure 2-1: Mars Science Laboratory sky-crane entry, decent, and landing sequence [12].

and attitude knowledge, followed by imminent mission loss due to a hazardous landing.

Although approximate belief state enumeration is applicable to nearly any embedded system, the MSL EDL demand for autonomous, responsive, and accurate monitoring and fault diagnosis, makes the decent stage IMU a highly relevant example that will be used throughout this thesis. This chapter presents a decent stage IMU system, similar to that of MSL, that is greatly simplified for pedagogical clarity, but sufficiently complex to highlight the innovation and importance of the additional accuracy provided by approximate belief state enumeration. The chapter concludes with graphical models of the IMU and its accessory components.

## 2.1 Simple IMU Example

An Inertial Measurement Unit (IMU) is a standard sensor package that is used to provide spacecraft and other robotic systems with translational and angular motion measurements. Space qualified IMUs are typically radiation hardened and tightly coupled to the spacecraft body; firmly mounted to the rigid-body structure, thermally controlled, and electrically wired to provide two way communication between the IMU and the attitude determination and control (ADC) processor. The IMU system modeled in this thesis is greatly simplified to three interconnected components, as shown in Figure 2-2.

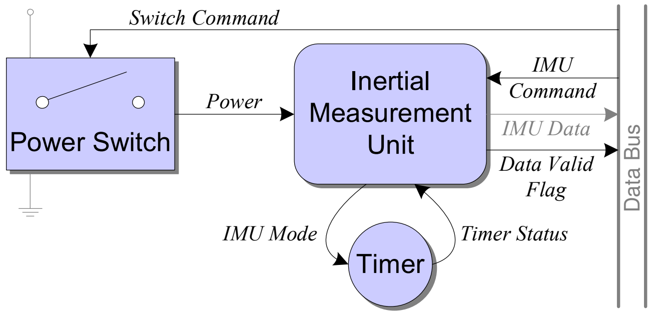


Figure 2-2: Simple IMU system block diagram.

This simplified IMU system consists of the basic IMU itself, a controllable Power Switch (PS) that provides the IMU with a power source, and a Timer (T) that is used to help infer whether the IMU has become stuck in an undesirable mode. Although the IMU and PS are interconnected with the data bus, which passes information to the ADC processor, we chose to not model this component for simplicity, and we assume that there is always power flowing into the Power Switch. Detailed descriptions of each component are provided below.

### 2.1.1 The IMU Component Model

This simple IMU is a relatively passive device that reacts primarily to the input power coming from the Power Switch. During nominal use, the IMU is ready to take measurements shortly after an initialization period, which occurs when it is first powered on. Although the main function of the IMU is to provide continuous motion measurements, for the purpose of monitoring and diagnosing the IMU operational mode, we will ignore this data. A graphical representation of the discrete IMU modes are shown in Figure 2-3.

The IMU has 5 defined operational modes, 3 of which are considered nominal and

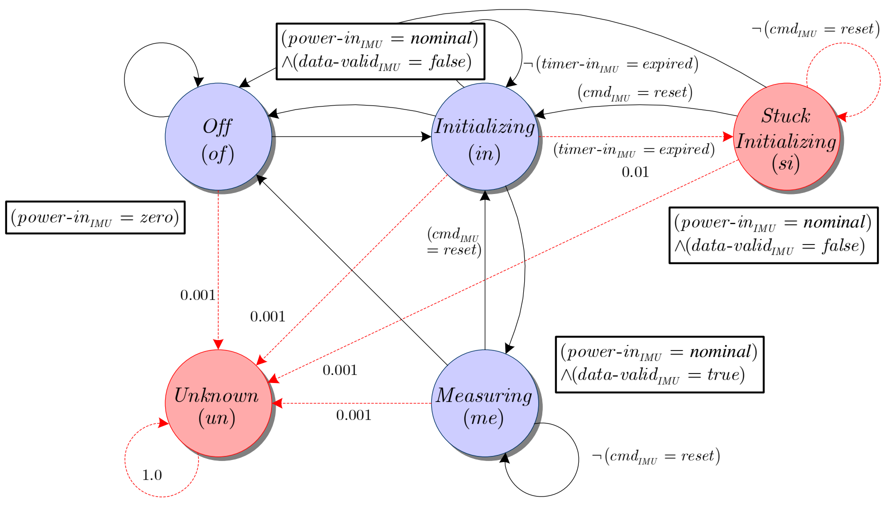


Figure 2-3: Model of IMU operational modes.

2 failure modes[1]. When the IMU first receives power, it transitions from the *Off* mode and begins initializing, during which the IMU measurements are not reliable and the *data-valid* flag is false. After a short period of time (typically less than 22 seconds), the IMU produces valid data and is ready for use. If the power is removed from the IMU at any point in time, it will return to the *Off* mode. It is also possible for the IMU to become *Stuck Initializing* if the Timer expires during the initialization process. This is a recoverable fault mode in which a reset command can be issued to to restart the IMU initialization. It is important to note that, if the IMU has to be reset multiple times, or there are unexplainable sensor measurements of the data validity that contradict the expected IMU behavior, it is more likely that the IMU is in an *Unknown* mode and the spacecraft should quickly switch to its redundant IMU (not modeled) in order to recover. The probabilities on the transitions will be explained at the end of this chapter in Section 2.2.

[1]For the purpose of this thesis, a failure mode is simply defined as an undesirable mode that is unexpected and rarely occurs (lower probability of occurrence).

### 2.1.2 The Power Switch Component Model

The sole purpose of the Power Switch (PS) is to provide the IMU with power. For the purpose of this simple IMU system, we will assume that the PS is always receiving power. An illustration of the PS is shown in Figure 2-4.

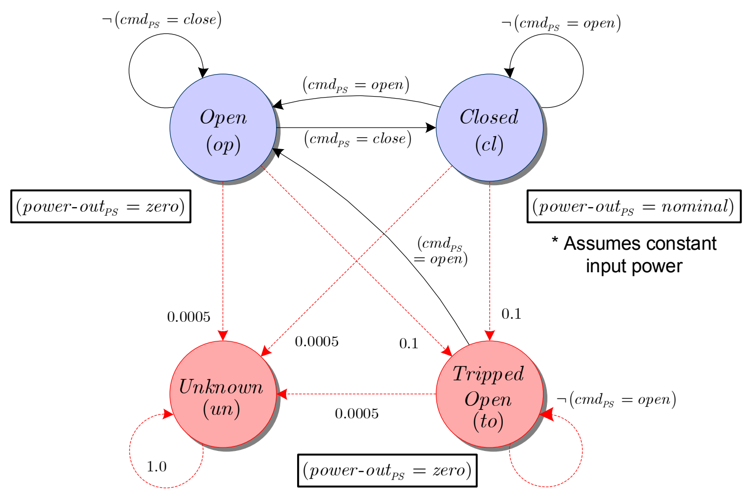


Figure 2-4: Model of Power Switch operational modes.

This Power Switch has 2 nominal modes and 2 failure modes. When the PS is commanded closed, the switch is shorted and power is supplied to the IMU. Likewise, when the PS is commanded open, power is removed from the IMU. In the presence of too much electron current, the PS will become *Tripped Open* in order to prevent overloading and possibly damaging the IMU. A safe recovery can be conducted by reopening the switch and then closing it again to power on the IMU. Similar to the IMU, there is also an *Unknown* mode, which captures all other unexpected behavior.

### 2.1.3 The Timer Component Model

The third component is the IMU Timer (T), shown in Figure 2-5, which is responsible for keeping track of how much time the IMU has spent *Initializing*. When the IMU is not initializing, the Timer is *Idle*. As soon as the IMU begins initializing, the Timer starts *Running* and an external continuous timer starts in the background. When the external timer expires after 22 seconds, the *alarm* flag is set to true, signaling that the IMU is

*Stuck Initializing*. Alternatively, if the IMU exits from the initializing mode before the external timer expires, the external timer will have no effect on this model and will simply reset the next time the IMU begins initializing.

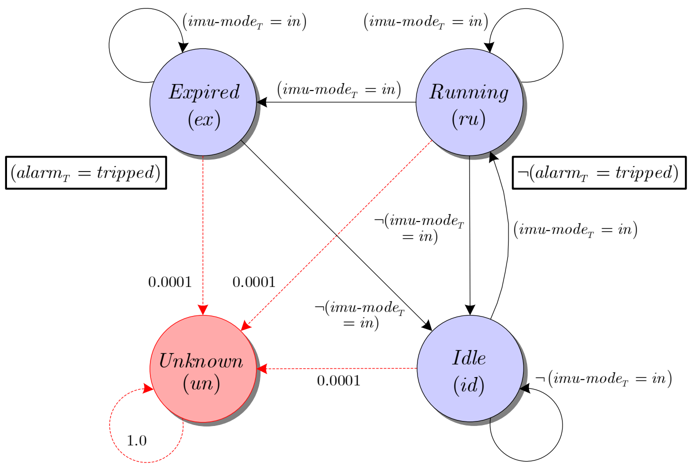

$(imu\text{-}mode_T = in)$
$(imu\text{-}mode_T = in)$
$(imu\text{-}mode_T = in)$
*Expired* (*ex*)
*Running* (*ru*)
$(alarm_T = tripped)$
$\neg(alarm_T = tripped)$
$\neg(imu\text{-}mode_T = in)$
$(imu\text{-}mode_T = in)$
0.0001
0.0001
$\neg(imu\text{-}mode_T = in)$
*Unknown* (*un*)
*Idle* (*id*)
0.0001
$\neg(imu\text{-}mode_T = in)$
1.0

Figure 2-5: Model of Timer operational modes.

This concludes our summary of the three components that constitute our simple pedagogical IMU plant. The next section section defines a specific type of factored HMM modeling formalism that is used in this thesis and provides a formal description of the simple IMU system. Chapter 3 introduces the complete PCCA estimation problem, followed by Chapter 4, describing the challenges that PCCA estimation presents for embedded systems and assumptions that have made monitoring and fault diagnosis tractable for full-scale systems.

## 2.2 PCCA Plant Model

As in previous work, we model the physical plant as a factored Hidden Markov Model that is compactly encoded as *Probabilistic Concurrent Constraint Automata (PCCA)* [20]. The PCCA represent a set of concurrently operating components that are interconnected and interact with their surrounding environment. Each automaton has a set of possible discrete modes with conditional probabilistic transitions, which capture both nominal and faulty behavior. These modes are only partially observable, due to a limited number of sensors, but are inherently constrained by the system properties that define each mode. In this section we review the formal definition of the PCCA plant model and provide an illustrative example using the simple IMU system that was previously introduced in Chapter 2.

### 2.2.1 PCCA Formalism

We first define a single probabilistic constraint automaton and then the composition of multiple automata. A probabilistic constraint automaton for component "$a$" is defined by the tuple $\mathcal{A}_a = \langle \Pi_a, \mathbb{M}_a, \mathbb{T}_a, \mathbf{P}_{\mathbb{T}_a} \rangle$:

1. $\Pi_a = \Pi_a^m \cup \Pi_a^r$ is a finite set of discrete variables for component "$a$", where each variable $\pi_a \in \Pi_a$ ranges over a finite domain $\mathbb{D}(\pi_a)$. $\Pi_a^m$ is a singleton set containing *mode* variable $\{x_a\} = \Pi_a^m$ whose domain $\mathbb{D}(x_a)$ is the finite set of discrete modes in $\mathcal{A}_a$. *Attribute* variables $\Pi_a^r$ include inputs, outputs, and any other variables used to define the behavior of the component. $\Sigma_a$ is the complete set of all possible full assignments over $\Pi_a$ and the state space of the component $\Sigma_a^{x_a} = \Sigma_a \Downarrow_{x_a}$ is the projection of $\Sigma_a$ onto mode variable $x_a$.

2. $\mathbb{M}_a : \Sigma_a^{x_a} \rightarrow \mathbb{C}(\Pi_a^r)$ maps each mode assignment $(x_a = v_a) \in \Sigma_a^{x_a}$ to a finite domain constraint $c_a(x_a = v_a) \in \mathbb{C}(\Pi_a^r)$, where $\mathbb{C}(\Pi_a^r)$ is the set of finite domain constraints over $\Pi_a^r$. These constraints are known as *modal constraints* and are typically encoded in the propositional form $\lambda \triangleq \textbf{True} \mid \textbf{False} \mid (u = y) \mid \neg\lambda_1 \mid \lambda_1 \wedge \lambda_2 \mid \lambda_1 \vee \lambda_2$, where $y \in \mathbb{D}(u)$. If the current mode is $(x_a^t = v_a)$ at time-step $t$, then the assignments to each attribute variable $r_a^t \in \Pi_a^r$ at time-step $t$ must be consistent with $c_a(x_a = v_a)$. These constraints capture the physical behavior of the mode.

3. $\mathbb{T}_a : \Sigma_a^{x_a} \times \mathbb{C}(\Pi_a^r) \rightarrow \Sigma_a^{x_a}$ is a set of transition functions. The set of finite domain constraints $\mathbb{C}(\Pi_a^r)$ are also known as the *transition guards*, encoded in the propositional form $\lambda$. Given a current mode assignment $(x_a = v_a) \in \Sigma_a^{x_a}$ and guard $g_a \in \mathbb{C}(\Pi_a^r)$ entailed at time-step $t$, each transition function $\tau_a(x_a = v_a, g_a) \in \mathbb{T}_a(x_a = v_a, g_a)$

specifies a target mode assignment $(x_a = v'_a) \in \Sigma_a^{x_a}$ that the automaton could transition into at time-step $t+1$. $\mathbb{T}_a = \mathbb{T}_a^n \cup \mathbb{T}_a^f$ captures both nominal and faulty behavior.

4. $\mathbf{P}_{\mathbb{T}_a} : \mathbb{T}_a(x_a = v_a, g_a) \rightarrow \Re[0,1]$ is a transition probability distribution. For each mode variable assignment in $\Sigma_a^{x_a}$ and guard $g_a^t$, there is a probability distribution across all transitions into target modes defined by the set of transition functions $\mathbb{T}_a(x_a = v_a, g_a)$.

The entire system plant $\mathcal{P}$ is modeled by a composition of concurrently operating constraint automata. Each automaton is interconnected to both its environment and other automata through constraints on shared variables. Formally, the PCCA plant model is defined by the tuple $\mathcal{P} = \langle \mathcal{A},\Pi,\mathbb{Q} \rangle$:

1. $\mathcal{A} = \{\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_n\}$ is the finite set of constraint automata that represent the $n$ components of the plant.

2. $\Pi = \bigcup_{a=1..n} \Pi_a$ is the set of all plant variables. The variables $\Pi$ are partitioned into a finite set of *mode* variables $\Pi^m = \bigcup_{a=1..n} \Pi_a^m$, *control* variables $\Pi^c \subseteq \bigcup_{a=1..n} \Pi_a^r$, *observation* variables $\Pi^o \subseteq \bigcup_{a=1..n} \Pi_a^r$, and *dependent* variables $\Pi^d \subseteq \bigcup_{a=1..n} \Pi_a^r$. $\Sigma^c$, $\Sigma^o$, and $\Sigma^d$ are the sets of full assignments over $\Pi^c$, $\Pi^o$, and $\Pi^d$.

3. $\mathbb{Q} \subset \mathbb{C}(\Pi)$ is a set of finite domain constraints that capture the interconnections between plant components.

### 2.2.2 Example: The IMU System PCCA Plant Model

The simple IMU system introduced in Chapter 2 will be used in this section to clarify the PCCA formalism. Our PCCA plant model $\mathcal{P}$ is composed of the constraint automata for the IMU ($\mathcal{A}_{imu}$), Power Switch ($\mathcal{A}_{ps}$), and Timer ($\mathcal{A}_t$). As an example of a single constraint automaton, consider the IMU component shown again in Figure 2-6.
A formal description of $\mathcal{A}_{imu}$ is provided below:

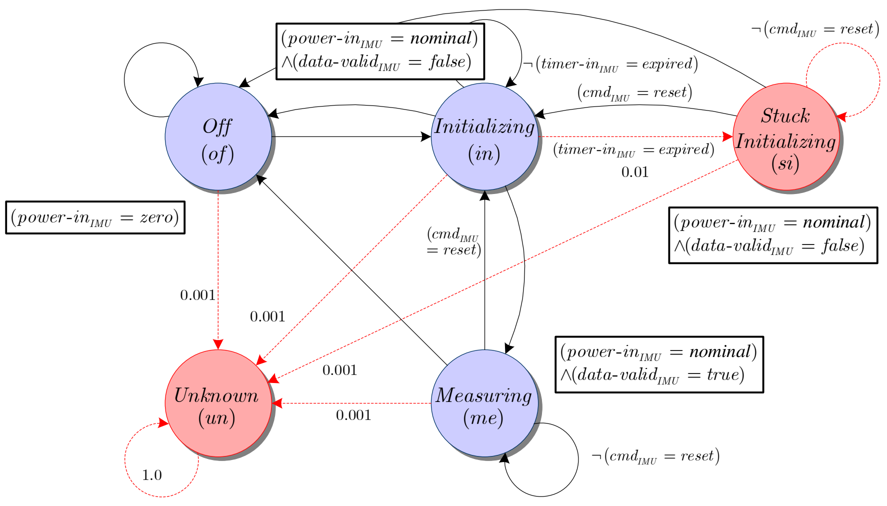

$(power\text{-}in_{IMU} = nominal)$
$\wedge(data\text{-}valid_{IMU} = false)$
$\neg(timer\text{-}in_{IMU} = expired)$
$\neg(cmd_{IMU} = reset)$
$(cmd_{IMU} = reset)$
Off (of)
Initializing (in)
Stuck Initializing (si)
$(timer\text{-}in_{IMU} = expired)$
0.01
$(power\text{-}in_{IMU} = zero)$
$(cmd_{IMU} = reset)$
$(power\text{-}in_{IMU} = nominal)$
$\wedge(data\text{-}valid_{IMU} = false)$
0.001
0.001
0.001
0.001
$(power\text{-}in_{IMU} = nominal)$
$\wedge(data\text{-}valid_{IMU} = true)$
Unknown (un)
Measuring (me)
1.0
$\neg(cmd_{IMU} = reset)$

Figure 2-6: IMU constraint automaton, $\mathcal{A}_{imu}$.

1. $\Pi_{imu} = \{x_{imu}, \mu^{cmd}_{imu}, o^{dv}_{imu}, d^{pi}_{imu}, d^{ti}_{imu}\}$ where $\{x_{imu}\} = \Pi^m_{imu}$ resides in 1 of 5 discrete modes $\mathbb{D}(x_{imu}) = \{of,\ in,\ me,\ si,\ un\}$ as indicated with circular nodes in Figure 2-6. $\Pi^r_{imu} = \{\mu^{cmd}_{imu}, o^{dv}_{imu}, d^{pi}_{imu}, d^{pi}_{imu}\}$ where $\mu^{cmd}_{imu}$ is used to reset the IMU with $\mathbb{D}(\mu^{cmd}_{imu}) = \{reset,\ no\text{-}command\}$, $o^{dv}_{imu}$ is an observation of the *data validity* with $\mathbb{D}(o^{dv}_{imu}) = \{true,\ false\}$, $d^{pi}_{imu}$ is the *power-in* with $\mathbb{D}(d^{pi}_{imu}) = \{zero,\ nominal\}$, and $d^{ti}_{imu}$ is a time expiration variable with $\mathbb{D}(d^{ti}_{imu}) = \{expired,\ not\text{-}expired\}$. $\Sigma_{imu} = \Sigma^m_{imu} \times \Sigma^r_{imu}$ is the set of all full assignments over $\Pi_{imu}$ with $5 \cdot (2 \cdot 2 \cdot 2 \cdot 2) = 80$ elements.

2. $\mathbb{M}_{imu}$ includes the constraints encapsulated by rectangles in Figure 2-6. The complete set of modal constraints for the IMU are shown in Table 2.1 below.

Table 2.1: $\mathcal{A}_{imu}$ Modal Constraints

| $(x_{imu} = v_{imu}) \in \Sigma^{x_{imu}}_{imu}$ | $\mathbb{M}_{imu}(x_{imu} = v_{imu})$ |
|---|---|
| $x_{imu} = of$ | $d^{pi}_{imu} = zero$ |
| $x_{imu} = in$ | $d^{pi}_{imu} = nominal \wedge o^{dv}_{imu} = false$ |
| $x_{imu} = me$ | $d^{pi}_{imu} = nominal \wedge o^{dv}_{imu} = true$ |
| $x_{imu} = si$ | $d^{pi}_{imu} = nominal \wedge o^{dv}_{imu} = false$ |
| $x_{imu} = un$ | (unconstrained) |

3. The component transitions are indicated by the arrows and labels in Figure 2-6. For example, $\mathbb{T}_{imu}(x_{imu} = si, \mu^{cmd}_{imu} = reset)$ is a set of transition functions

$\{\tau^{n_1}, \tau^{n_2}, \tau^{f_1}\}$ where $(x_{imu} = si)$ is the source mode and $(x_{imu} = in)$, $(x_{imu} = of)$, and $(x_{imu} = un)$ are the target modes. The complete set of transition functions for $\mathcal{A}_{imu}$ are shown in Table 2.2 below.

Table 2.2: $\mathcal{A}_{imu}$ Transition Functions / Probabilities

| $(x_{imu} = v_{imu})$ | $g_{imu} \in \mathbb{C}(\Pi^r_{imu})$ | $\mathbb{T}_{imu}(x_{imu} = v_{imu}, g_{imu})$ | $\mathbf{P}_{\mathbb{T}_{imu}}(x_{imu} = v_{imu}, g_{imu})$ |
|---|---|---|---|
| $x_{imu} = of$ | (unconstrained) | $\{of, in, un\}$ | $\{0.4995, 0.4995, 0.001\}$ |
| $x_{imu} = in$ | $\neg(d^{ti}_{imu} = expired)$ | $\{of, in, me, un\}$ | $\{0.333, 0.333, 0.333, 0.001\}$ |
| $x_{imu} = in$ | $d^{ti}_{imu} = expired$ | $\{of, me, si, un\}$ | $\{0.4945, 0.4945, 0.01, 0.001\}$ |
| $x_{imu} = me$ | $\neg(\mu^{cmd}_{imu} = reset)$ | $\{of, me, un\}$ | $\{0.4995, 0.4995, 0.001\}$ |
| $x_{imu} = me$ | $\mu^{cmd}_{imu} = reset$ | $\{of, in, un\}$ | $\{0.4995, 0.4995, 0.001\}$ |
| $x_{imu} = si$ | $\neg(\mu^{cmd}_{imu} = reset)$ | $\{of, si, un\}$ | $\{0.4995, 0.4995, 0.001\}$ |
| $x_{imu} = si$ | $\mu^{cmd}_{imu} = reset$ | $\{of, in, un\}$ | $\{0.4995, 0.4995, 0.001\}$ |
| $x_{imu} = un$ | (unconstrained) | $\{un\}$ | $\{1\}$ |

4. The component transition probability distribution for each set of IMU transition functions is shown on the right side of Table 2.2 above.

For completeness, the formal definitions to $\mathcal{A}_{ps}$ and $\mathcal{A}_t$ are listed in Appendix A. The full PCCA plant model for this simple IMU system is composed of 3 components and the interconnections between them. The power output of the PS is connected to the power input of the IMU and the T is connected to the IMU to help determine if the IMU has become *Stuck Initializing*[2]. The PCCA plant $\mathcal{P}$ for the IMU system is formally defined as follows:

1. $\mathcal{A} = \{\mathcal{A}_{imu}, \mathcal{A}_{ps}, \mathcal{A}_t\}$ is the set of all constraint automata in the IMU system; including the IMU, Power Switch, and Timer.

2. $\Pi = \Pi_{imu} \cup \Pi_{ps} \cup \Pi_t$ is the set of all variables. This set is partitioned into *mode*

[2]The transient initialization process of the IMU is a perfect example of when a Timed Plant Model [13] would be preferred over a PCCA to increase the fidelity of the model. Since this thesis is limited to only PCCA models, we have compensated for the continuous-time IMU behavior by introducing a discrete *Initializing* mode for the IMU as well as a Timer component, that interacts with an external continuous timer, to help determine if the IMU is *Stuck Initializing.* Although the concepts of Best-First Belief State Enumeration in this thesis are presented in the context of PCCA, the method could be expanded to improve the accuracy of Timed Mode Estimation.

variables $\Pi^m = \{x_{imu}, x_{ps}, x_t\}$, *control* variables $\Pi^c = \{\mu_{imu}^{cmd}, \mu_{ps}^{cmd}\}$, *observation* variables $\Pi^o = \{o_{imu}^{dv}, o_t^{al}\}$, and *dependent* variables $\Pi^d = \{d_{imu}^{pi}, d_{imu}^{ti}, d_{ps}^{po}, d_t^{im}\}$.

3. The interconnections for the IMU system are

$$\mathbb{Q} = \left\{ \begin{array}{r} d_{ps}^{po} = d_{imu}^{pi} \wedge \\ x_{imu} = d_t^{im} \wedge \\ x_t = d_{imu}^{ti} \end{array} \right\},$$

where the *power-out* (*po*) of the PS is connected to the *power-in* (*pi*) of the IMU, the IMU mode variable $x_{imu}$ is connected to *imu-mode* (*im*) of the T, and the T mode variable $x_t$ is connected to the *timer-in* (*ti*) of the IMU.

With the PCCA formalism defined, the next chapter will introduce the full PCCA estimation problem as well as approximations that are used to make the problem scalable to full-sized systems.

# Chapter 3

# Estimation of PCCA

This chapter reviews exact belief state update for PCCAs and presents a formulation of the estimation problem as an Optimal Constraint Satisfaction Problem (OCSP), interleaving examples using the IMU system. The task of estimation is to calculate a belief state of the system in real-time, while maintaining accuracy and reliability. A belief state is a probability distribution over the states of a system, which represents the likelihood of the system being in any single state, given a history of past commands and observations. For PCCA, a state $s_i$ is defined as a full assignment to mode variables $s_i \in \Sigma^m$ and a belief state $B = \langle S, p \rangle$ is a finite set of estimates that cover all consistent states $S \subseteq \Sigma^m$. Each estimate consists of a state $s_i \in S$ and its posterior probability $p(s_i) \in p$.

## 3.1 Belief State Update

The Markov property declares that the future state of a system is conditionally independent of its past, given its current belief state. This property allows an estimator to iteratively compute the next complete belief state $B^{t+1}$ at time-step $t+1$ by only considering the current belief state $B^t$ and commands $\mu^t$ at time-step $t$, along with the resulting observations $o^{t+1}$. The belief state is then computed using the standard HMM belief state update equations [1]:

$$\mathbf{P}(s_j^{t+1}|o^{<0,t>}, \mu^{<0,t>}) = \sum_{s_i^t \in S^t} \left(\mathbf{P}(s_j^{t+1}|s_i^t, \mu^t)\mathbf{P}(s_i^t|o^{<0,t>}, \mu^{<0,t-1>})\right) \tag{3.1}$$

$$\mathbf{P}(s_j^{t+1}|o^{<0,t+1>}, \mu^{<0,t>}) = \frac{\mathbf{P}(s_j^{t+1}|o^{<0,t>}, \mu^{<0,t>}) \cdot \mathbf{P}(o^{t+1}|s_j^{t+1})}{\sum_{s_i^{t+1} \in S^{t+1}} \mathbf{P}(s_i^{t+1}|o^{<0,t>}, \mu^{<0,t>})\mathbf{P}(o^{t+1}|s_i^{t+1})} \tag{3.2}$$

Equation 3.1 represents the *a priori* probability of being in the next state $s_j^{t+1}$ at time-step $t+1$, given all the observations $o^{<0,t>}$ and commands $\mu^{<0,t>}$ between time-step 0 and $t$. $\mathbf{P}(s_i^t|o^{<0,t>}, \mu^{<0,t-1>}) \in p^t$ is the probability that the system was in state $s_i$ at time-step $t$ and $\mathbf{P}(s_j^{t+1}|s_i^t, \mu^t)$ is the state transition probability. Equation 3.1 propagates the system dynamics into the future before considering new observations. Once all the *a priori* estimates are generated, Equation 3.2 then updates these estimates by adjusting the probabilities based on new observations $o^{t+1}$ using the Total Probability Theorem and Bayes' Rule to calculate the *a posteriori* probabilities $p^{t+1}$ across all states in $S^{t+1}$.

The belief state evolution over time can be visualized in a *Trellis diagram*, as shown in Figure 3-1. Each column represents a separate belief state at different time-steps. Arrows depict conditional dependence between states, and correspond to transitions between states. The probabilities associated with each state in the belief states are not shown.

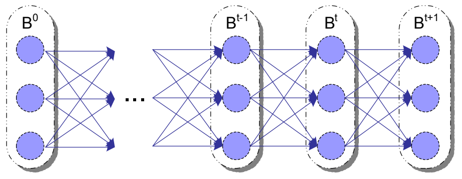


Figure 3-1: The *Trellis diagram* of the possible state evolutions over time.

### 3.1.1 PCCA Transition and Observation Probabilities

To complete the definition of belief state update for PCCA, the state transition probabilities and observation probabilities must be defined. Since the PCCA are concurrently operating, the state transition consists of a set of component mode transitions; one mode transition $\tau_a$ for each component $(x_a = v_a) \in s_i$. By assuming that each component transition is conditionally independent given the current state $s_i^t$ and commands $\mu^t$, the state transition probability simply becomes the product of the component transition probabilities (Equation 3.3). This assumption, previously made by Livingstone [21], has been demonstrated in practice to be reasonable for a wide range of engineered systems.

$$\mathbf{P}(s_j^{t+1}|s_i^t, \mu^t) = \prod_{(x_a^{t+1}=v_a')\in s_j^{t+1}} \left(\mathbf{P}(x_a^{t+1} = v_a'|x_a^t = v_a, s_i^t, \mu^t)\right) \tag{3.3}$$

The component mode transition probability $\mathbf{P}(x_a^{t+1} = v_a'|x_a^t = v_a, s_i^t, \mu^t)$ is the probability of transitioning from mode $(x_a^t = v_a)$ at time-step $t$ to mode $(x_a^{t+1} = v_a')$ at the next time-step, conditioned on the current state of all components $s_i^t$ and commands $\mu^t$. Recall from the PCCA formalism that if transition guards $g_a$ are entailed by $s_i^t$ and $\mu^t$, the set of transitions $\mathbb{T}_a(x_a = v_a, g_a)$ are considered to be *enabled* and their target modes are *reachable*; otherwise the transitions are disabled with a transition probability of zero[1]. To be probabilistically complete, the sum of the enabled outgoing state transition probabilities must be 1 for each $s_i^t \in B^t$.

The conditional observation probability $\mathbf{P}(o^{t+1}|s_j^{t+1})$ is the probability of sensing observations $o \in \Sigma^o$, given that the system is in state $s_j \in \Sigma^m$ at time-step $t+1$. For PCCA, the observation probability distribution is defined using a consistency approach similar to that of GDE [6], such that for every state $s_j \in \Sigma^m$, there is a probability distribution across all combinations of observations. If every observation $o_l \in o^{t+1}$ is entailed or refuted by the conjunction of the modal constraints $\mathbb{M}$ and state $s_j^{t+1}$, the observation probability $\mathbf{P}(o^{t+1}|s_j^{t+1})$ is 1 or 0, respectively. When the observations are neither entailed nor refuted, there is a uniform probability distribution of $1/m$ across all the $m$ possible consistent values of $o^{t+1}$, creating a probabilistic bias towards states that predict (entail) observations. This uniform distribution assumption is a degenerate case of Maximum-Entropy [14] when there is no previous knowledge about how the sensors behave. The precise observation probability distribution for PCCA is shown in Equation 3.4.

$$\mathbf{P}(o^{t+1}|s_j^{t+1}) = \begin{cases} 1 & \text{if } s_j^{t+1} \wedge \mathbb{M} \models o^{t+1}, \\ 0 & \text{if } s_j^{t+1} \wedge \mathbb{M} \models \neg o^{t+1}, \\ 1/m & \text{otherwise,} \end{cases} \tag{3.4}$$

where $m$ = number of consistent assignments to $o^{t+1}$ for $s_j^{t+1}$ and $\mathbb{M}$.

[1]On rare occasion, it is possible to have a transition guard that is neither entailed nor refuted, resulting in a probability that the transition is enabled. This is beyond the scope of this thesis but is discussed as future work in Section 6.2.

### 3.1.2 Example: IMU System Belief State Update

The simple IMU system described in Section 2.1, and formally defined in Section 2.2, is now used to demonstrate the mechanics of the propagate and update equations that were introduced in Section 3.1. This example uses the belief state update equations to compute the exact solutions to the PCCA estimation problem. Figure 3-2 illustrates a single estimation cycle where the belief state update equations first propagate the IMU system dynamics to calculate the *a priori* state probabilities (center of Figure 3-2) and then update those estimates with the resulting observation data to determine the *a posteriori* state probabilities (right of Figure 3-2). The labels on the arrows represent the component transition probabilities (one for each component) during the propagation step, and observation probabilities during the update step. Due to the large state space of this simple example[2], only the leading four estimates are shown for $B^{t+1}$.

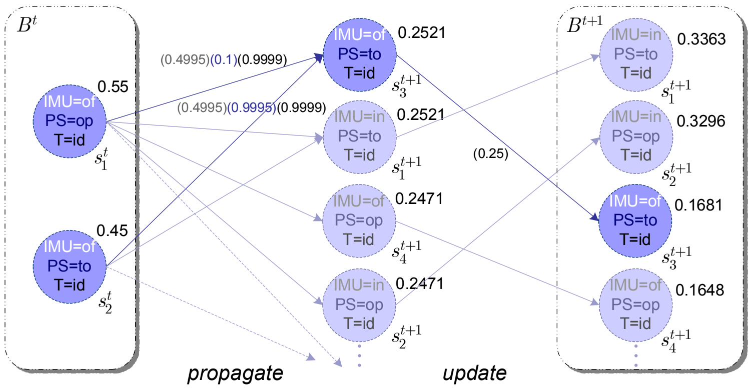


Figure 3-2: Single-step exact belief state update of the IMU Plant.

Given the current belief state $B^t$, consisting of $s_1^t$ with probability 0.55 and $s_2^t$ with probability 0.45, we will focus on the process of generating the state probability for $s_3^{t+1}$, as highlighted in Figure 3-2. Assuming that there are no commands $\mu^t$ and only consistent

[2]Since all of the components are independently operating, the actual number of estimates contained in the full belief state is $\prod_{y_k \in \Pi^m} |\mathbb{D}(y_k)|$, where $y_k$ contains only the *reachable* mode assignments of $x_k^{t+1}$ such that $\mathbb{D}(y_k) \subseteq \mathbb{D}(x_k^{t+1})$. Recall that a mode is *reachable* if there is an enabled component transition leading to that mode. For this example, the number of elements in the belief state $B^{t+1}$ is $(3)(3)(2) = 18$.

observations $o^{t+1}$, both current states have enabled state transitions that converge to a single next state $s_3^{t+1}$ with an *a priori* probability of 0.2521. This state is then updated with its observation probability to result in an *a posteriori* state probability of 0.1618. Before we can calculate the *a priori* state probabilities, we must first determine all the enabled component mode transitions by determining which transition guards are entailed. The enabled transitions and their probabilities for all three components of the IMU system are shown in Table 3.1 below:

Table 3.1: Enabled Component Transitions for the IMU Plant, $\mathcal{P}$

| $s^t$ | $(x_{imu} = v_{imu})$ | $g_{imu}$ | $\mathbb{T}_{imu}$ | $\mathbf{P}_{\mathbb{T}_{imu}}$ |
|---|---|---|---|---|
| $s_1^t$ | $x_{imu} = of$ | (unconstrained) | $\{of, in, un\}$ | $\{0.4995, 0.4995, 0.001\}$ |
| $s_2^t$ | $x_{imu} = of$ | (unconstrained) | $\{of, in, un\}$ | $\{0.4995, 0.4995, 0.001\}$ |

| $s^t$ | $(x_{ps} = v_{ps})$ | $g_{ps}$ | $\mathbb{T}_{ps}$ | $\mathbf{P}_{\mathbb{T}_{ps}}$ |
|---|---|---|---|---|
| $s_1^t$ | $x_{ps} = op$ | $\neg(\mu_{ps}^{cmd} = close)$ | $\{op, to, un\}$ | $\{0.8995, 0.1, 0.0005\}$ |
| $s_2^t$ | $x_{ps} = to$ | $\neg(\mu_{ps}^{cmd} = open)$ | $\{to, un\}$ | $\{0.9995, 0.0005\}$ |

| $s^t$ | $(x_t = v_t)$ | $g_t$ | $\mathbb{T}_t$ | $\mathbf{P}_{\mathbb{T}_t}$ |
|---|---|---|---|---|
| $s_1^t$ | $x_t = id$ | $\neg(d_t^{im} = in)$ | $\{id, un\}$ | $\{0.9999, 0.0001\}$ |
| $s_2^t$ | $x_t = id$ | $\neg(d_t^{im} = in)$ | $\{id, un\}$ | $\{0.9999, 0.0001\}$ |

Now we can compute the state transition probabilities by taking the product of the enabled component transition probabilities. Since we are interested in state $s_3^{t+1}$, we will only consider the state transitions leading to the target state $\{x_{imu} = of, x_{ps} = to, x_t = id\}$. The probabilities for the two state transitions that converge to state $s_3^{t+1}$ are computed below:

$$\begin{aligned}
\mathbf{P}(s_3^{t+1}|s_1^t, \mu^t) &= \mathbf{P}(x_{imu} = of|x_{imu} = of)\mathbf{P}(x_{ps} = to|x_{ps} = op)\mathbf{P}(x_t = id|x_t = id) \\
&= (0.4995)(0.1)(0.9999) \\
&= 0.049945 \\
\mathbf{P}(s_3^{t+1}|s_2^t, \mu^t) &= \mathbf{P}(x_{imu} = of|x_{imu} = of)\mathbf{P}(x_{ps} = to|x_{ps} = to)\mathbf{P}(x_t = id|x_t = id) \\
&= (0.4995)(0.9995)(0.9999) \\
&= 0.4992
\end{aligned}$$

Using the belief state update propagate equation from Equation 3.1, we can now calculate the *a priori* state probability by multiplying each state transition probability by their originating state probability and then summing over all the incoming transitions into state $s_3^{t+1}$.

$$\begin{aligned}\mathbf{P}(s_3^{t+1}|o^{<0,t>},\mu^{<0,t>}) &= \mathbf{P}(s_3^{t+1}|s_1^t,\mu^t)\mathbf{P}(s_1^t|o^{<0,t>},\mu^{<0,t-1>}) \\ &\quad + \mathbf{P}(s_3^{t+1}|s_2^t,\mu^t)\mathbf{P}(s_2^t|o^{<0,t>},\mu^{<0,t-1>}) \\ &= (0.049945)(0.55) + (0.4992)(0.45) \\ &= 0.2521\end{aligned}$$

This is the same *a priori* state probability shown in Figure 3-2. The next step is to update this probability with the resulting observations $o^{t+1}$ using Equation 3.2. In this example, we have assumed that we received observations that are consistent with $s_3^{t+1}$. By reconciling the modal constraints of the IMU automata defined in Chapter 2.2 and Appendix A, there are four unique sets of observation assignments that are consistent with both $s_3^{t+1}$ and $\mathbb{M}$:

$$\begin{aligned}&\{o_{imu}^{dv} = true, o_t^{al} = tripped\}, \{o_{imu}^{dv} = true, o_t^{al} = not\text{-}tripped\}, \\ &\{o_{imu}^{dv} = false, o_t^{al} = tripped\}, \{o_{imu}^{dv} = false, o_t^{al} = not\text{-}tripped\}.\end{aligned}$$

Since a set of observations is neither entailed nor refuted, the observation probability for state $s_3^{t+1}$ is $\mathbf{P}(o^{t+1}|s_3^{t+1}) = 1/4 = 0.25$ (as shown in Figure 3-2), where there are 4 consistent sets of observations. This observation probability is then used in Equation 3.2 to compute the *a posteriori* probability as shown below:

$$\begin{aligned}\mathbf{P}(s_3^{t+1}|o^{<0,t+1>},\mu^{<0,t>}) &= \frac{\mathbf{P}(s_3^{t+1}|o^{<0,t>},\mu^{<0,t>}) \cdot \mathbf{P}(o^{t+1}|s_3^{t+1})}{\sum_{s_i^{t+1}\in S^{t+1}} \mathbf{P}(s_i^{t+1}|o^{<0,t>},\mu^{<0,t>})\mathbf{P}(o^{t+1}|s_i^{t+1})} \\ &= \frac{(0.2521)(0.25)}{(0.37487)} \\ &= 0.1681\end{aligned}$$

The denominator of Equation 3.2 is a normalization factor that can be acquired by summing over the all posterior probabilities for belief state $B^{t+1}$ prior to normalization. For this example, the normalization factor is 0.37487.

This completes the belief state update for state $s_3^{t+1}$. In order to compute the full belief state, this same process must be completed for all 18 possible reachable states in $B^{t+1}$. It is important to note that although $s_1^{t+1}$ and $s_2^{t+1}$ are more likely than $s_3^{t+1}$ in $B^{t+1}$ (as illustrated in Figure 3-2), the IMU cannot be *Initializing* while the Power Switch is *Tripped Open* or *Open*, since the Power Switch does not supply the IMU with power in either of those modes[3]. Due to our factored model representation, belief state update alone will not eliminate these inconsistent states. A solution to this problem is presented in the next section by framing the estimation problem as an Optimal Constraint Satisfaction Problem (OCSP) [22]. Using this framework, a solution is only valid if all the constraints are satisfied, hence, inconsistent states $s_1^{t+1}$ and $s_2^{t+1}$ are invalid solutions and would not be returned.

## 3.2 Optimal Constraint Satisfaction Problems

PCCA estimation can be viewed as a problem of constraint optimization, where each reachable target state $s_j^{t+1}$ in the belief state $B^{t+1}$ must be consistent with modal constraints $\mathbb{M}$, component interconnections $\mathbb{Q}$, and observations $o^{t+1}$. This constraint optimization formulation was previously used in Titan [20] and can similarly be used to formulate the methods underlying GDE [6], Sherlock [7], and Livingstone [21, 15]. This thesis leverages a similar OCSP formulation, but differs from previous approaches by augmenting the utility function specification to increase the estimator accuracy.

**Definition 3.1.** *An OCSP $\langle \boldsymbol{y}, f, C \rangle$ is a problem of the form "arg max $f(\boldsymbol{x})$ subject to $C(\boldsymbol{y})$," where $\boldsymbol{x} \subseteq \boldsymbol{y}$ is a vector of decision variables, $C(\boldsymbol{y})$ is a set of state constraints, and $f(\boldsymbol{x})$ is a multi-attribute utility function.*

[3]The HMM belief state update equations enumerated inconsistent states because our IMU Plant model has violated the conditional independence assumption of the component transition probabilities, and not because of a flaw in the belief state update equations. In these circumstances, there is actually a joint probabiliy distribution across the enabled transitions. A couple possible solutions to this problem are presented in the future work section on Page 82.

Solving an OCSP consists of generating a prefix to the sequence of feasible solutions, ordered by decreasing value of $f$. A feasible solution assigns to each variable in $\mathbf{x}$ a value from its domain, such that $C(\mathbf{y})$ is satisfied. For PCCA estimation, the decision variables $\mathbf{x}$ are the set of mode variables $\Pi^m$ and the constraints $C(\mathbf{y})$ restrict mode variable assignments $(x_a = v'_a)$ to those that are consistent with observations $o^{t+1}$, modal constraints $\mathbb{M}_a(x_a = v'_a)$, and component interconnections $\mathbb{Q}$. Algorithm 3.1 provides pseudo code for computing the exact belief state update when framing PCCA estimation as an OCSP.

**Algorithm 3.1** $\textsc{BeliefStateUpdate}(\mathcal{P}, B^t, \mu^t, o^{t+1})$

```
1: Setup the OCSP $\langle \mathbf{y}, f, C \rangle$:
     • The vector $\mathbf{x}$ includes a decision variable $\mathbf{x}_a$ for each component of the plant, whose domain $\mathbb{D}(\mathbf{x}_a)$ is the set of modes that are reachable from any current state $s_i^t \in S^t$. For all $s_i^t \in S^t$, the target mode for each transition $(x_a = v'_a) = \tau_a(x_a = v_a, g_a)$ whose source $(x_a = v_a) \in s_i^t$ and guard $g_a$ are satisfied by $C_M^t \wedge s_i^t \wedge \mu^t$ is considered reachable, such that $v'_a \in \mathbb{D}(\mathbf{x}_a)$. $C_M^t = \mathbb{Q} \wedge (\bigwedge_{(x_a = v_a) \in s_i^t} \mathbb{M}_a(x_a = v_a))$.
     • The utility function $f(\mathbf{x})$ is the posterior probability of next state $\mathbf{x}$. More precisely, $f(\mathbf{x}) = \left(\sum_{s_i^t \in S^t} \mathbf{P}(\mathbf{x} \mid s_i^t, \mu^t) \cdot p^t(s_i)\right) \cdot \mathbf{P}(o^{t+1} \mid \mathbf{x})$, where $\mathbf{P}(\mathbf{x} \mid s_i^t, \mu^t) = \prod_{(x_a = v'_a) \in \mathbf{x}} \mathbf{P}(x_a = v'_a \mid x_a = v_a, s_i^t, \mu^t)$, $p^t(s_i)$ is the posterior probability for state $s_i^t$, and $\mathbf{P}(o^{t+1} \mid \mathbf{x})$ is the observation probability for $\mathbf{x}$.
     • $C(\mathbf{y})$ encodes the constraint that $\mathbf{x} \wedge C_{M_\mathbf{x}} \wedge o^{t+1}$ must be consistent. $C_{M_\mathbf{x}} = \mathbb{Q} \wedge (\bigwedge_{(x_a = v'_a) \in \mathbf{x}} \mathbb{M}_a(x_a = v'_a))$.
2: Compute all the solutions $S^{t+1}$ to OCSP$\langle \mathbf{y}, f, C \rangle$.
3: Extract the normalized posterior state estimate probabilities, such that $p^{t+1}(s_j) = f(s_j) / \sum_{s_i \in S^{t+1}} f(s_i)$ for each solution $s_j \in S^{t+1}$.
4: return the consistent state estimates contained by $B^{t+1} = \langle S^{t+1}, p^{t+1} \rangle$.
```

Algorithm 3.1 first initializes the OCSP in Step 1 with the current belief state $B^t$, commands $\mu^t$, and resulting sensor observations $o^{t+1}$. All the consistent states in the next belief state $B^{t+1}$ are then computed and stored in $S^{t+1}$ in Step 2. The posterior probabilities $p^{t+1}$ are then computed in Step 3 by taking the utility function of Step 1 and normalizing across all states $s_j^{t+1} \in S^{t+1}$, as per the HMM update equation (recall Equation 3.2 on page 29). This procedure is repeated for each estimation cycle.

The challenging part of Algorithm 3.1 is in computing the solutions to the OCSP in Step 2. Solutions can be computed using any OCSP solver, but this thesis will focus on using OPSAT as an efficient OCSP solver that generates solutions in order of likelihood.

Chapter 4 describes OPSAT and gives justification for why best-first enumeration is a key property for reactive estimation.

### 3.2.1 Example: IMU System OCSP Belief State Update

Recall the single-step belief state update example for the IMU system shown in Figure 3-2 and represented more compactly in Figure 3-3a. By framing PCCA estimation as an OCSP and computing the belief state using Algorithm 3.1, the resulting belief state $B^{t+1}$ is shown in Figure 3-3b. The only difference is that the leading two estimates of $B^{t+1}$ in Figure 3-3a were correctly determined by Algorithm 3.1 to be inconsistent and were not returned as valid solutions; elevating $s_3^{t+1}$ to $s_1^{t+1}$ in Figure 3-3b. The difference in estimate probabilities is due to normalization over only the states that are consistent[4].

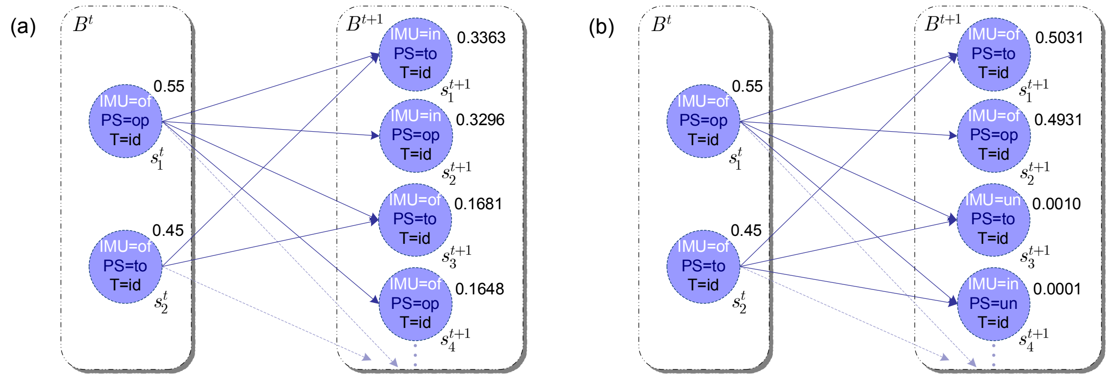


Figure 3-3: Single-step belief state update example for the IMU Plant (a) solved without considering constraints and (b) solved as an OCSP that considers constraints.

The next chapter begins by identifying 3 approximations that have previously made online estimation tractable when scaled up to full-sized systems. The two main contributions of this thesis are then presented as improvements to existing mode estimation

[4]Although framing PCCA estimation as an OCSP correctly eliminates the inconsistent states, it introduces an error in the outgoing state transition probability distribution, such that the sum of probabilities across the outgoing state transitions is no longer 1. For example, since the transitions from $s_1^t$ to $s_1^{t+1}$ and $s_2^{t+1}$ in Figure 3-3a no longer exist as a solution to the OCSP, the sum of the outgoing state transition probabilities from $s_1^t$ is $1-(0.4995)(0.1)(0.9999)-(0.4995)(0.8995)(0.9999) = 0.5008 \neq 1$. This is because our assumption of independent transitions is occasionally violated due to the interconnected nature of the components. One solution to this problem is to normalize the outgoing state transitions during the estimation process but this approach is not reflected in Algorithm 3.1.

approaches that, together, eliminate 2 of the 3 approximations. These two novel estimation techniques, known as *Best-First Belief State Enumeration (BFBSE)*[16] and *Best-First Belief State Update (BFBSU)*, provide a highly accurate mode estimation capability without incurring much additional computational overhead.

# Chapter 4

# Approximate Estimation of PCCA

As space exploration systems grow increasingly complex, there is an unprecedented demand for accurate and reactive monitoring and diagnosis techniques that must be scalable to mounting challenges. Due to limited computational resources of embedded systems, the exact solution presented in Chapter 3 is not tractable for full-sized systems. This chapter begins by identifying three key approximations that were previously used to achieve reactivity, while preserving estimator accuracy. Section 4.2 introduces an OCSP solver that efficiently solves the *approximate* estimation problem using *Conflict-directed* $A^*$ [22, 18] and Sections 4.3 and 4.4 adjust the $\text{A}^*$ heuristic function to eliminate two of the three belief state update approximations.

## 4.1 Belief State Update Approximations

Three significant approximations are made by previous monitoring and diagnosis engines; including Livingstone [21, 15], and previously in Titan [20]:

1. The full belief state $B^t$ is accurately approximated by maintaining only the $k$ most likely estimates in an approximate belief state $\tilde{B}^t$.

2. The probability of each state is accurately approximated by the probability of the most likely trajectory to that state.

3. The observation probabilities can be accurately reduced to 1.0 for all observations consistent with the state, and 0.0 for observations inconsistent with the state.

This thesis continues to make Approximation 1 to mitigate state space explosion, but eliminates Approximations 2 and 3. The following section present each approximation in detail.

### 4.1.1 Exponential Belief State Approximation

For systems modeled as PCCA, there is a finite number of estimates in the belief state, though the size of the belief state is exponential in the number of concurrently operating components. More precisely, size of the belief state for $n$ components is $\prod_{a=1..n} |\mathbb{D}(x_a)|$, where $|\mathbb{D}(x_a)|$ is the number of modes in $\mathcal{A}_a$. For a full-scale spacecraft propulsion subsystem, such as the NewMaap model of JPL's Cassini Spacecraft propulsion subsystem, the size of the belief state was roughly $3.5^{80}$ (80 mode variables with an average domain size of 3.5) [21]. To mitigate this belief state space explosion, previous work on Livingstone and Titan have made the assumption that the true state of the system is captured within only a few of the most likely estimates. This assumption is based on the key insight that, although the full belief state is exponential in size, the bulk of the probability density is concentrated in only a handful of the most likely estimates. This is due to the drastically decreasing likelihood of simultaneous multiple point failures [7].

Recall the single-step exact belief state update OCSP example for the IMU plant, presented earlier in Section 3.2.1, and summarized again in Figure 4-1a. The resulting probability distribution across all 16 consistent states in $B^{t+1}$ is shown in Figure 4-1b, where the states are ordered in terms of likelihood. The leading 2 estimates capture 99.62% of the total belief state probability density, supporting the hypothesis that the true state of the system is likely to be contained within the leading most likely estimates.

By leveraging this approximation, the estimation problem is simplified from updating the full belief state $B$ to enumerating the $k$ best estimates in an approximate belief state $\tilde{B}$. In order to avoid extraneous computation, preserve reactivity of the estimation process, and enable it to be employed for the purposes of real-time control, this enumeration is performed in best-first order. Section 4.2 reviews an efficient technique for best-first estimate enumeration, based on *Conflict-directed* $A^*$ [22].

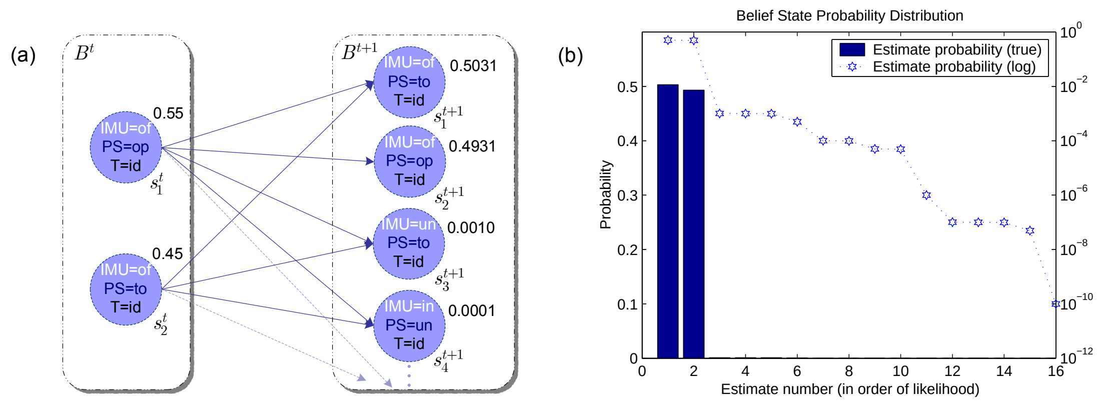


Figure 4-1: Belief state probability distribution at $B^{t+1}$ for the simple IMU Plant scenaio.

## 4.1.2 Trajectory Approximation

Although the majority of belief state probability density can be captured by only a handful of most likely estimates, it is not clear how to quickly identify which estimates, out of the entire exponential belief state, are most likely. Previous mode estimation approaches side-step this problem by unfolding the belief state transitions into a branching tree structure (Figure 4-2) and by enumerating estimates in order of state trajectory probability[1] [21, 15, 20].

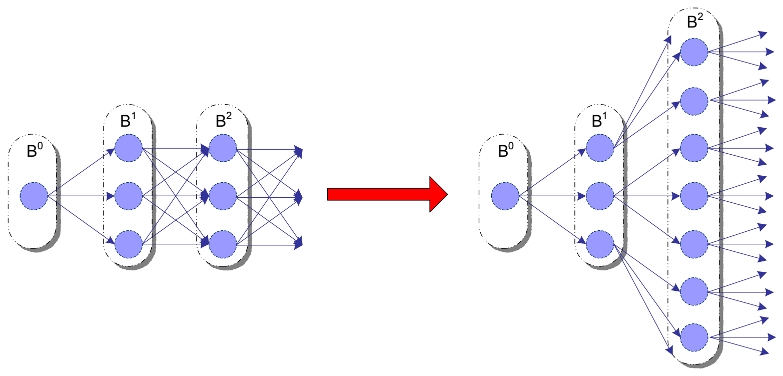


Figure 4-2: Evolution of the belief state, represented as a *trellis diagram* (left), can be decomposed into a *branching tree.*

Each arrow on the right side of Figure 4-2 still represents a state transition, but

[1]The state trajectory probability is defined as the product of state transition probability $\mathbf{P}(s_j^{t+1} \mid s_i^t, \mu^t)$ and its source state probability $\mathbf{P}(s_i^t|o^{<0,t>}, \mu^{<0,t-1>})$.

the belief state encoding is no longer compact since there may be duplicate states at each time-step in the branching tree. Furthermore, the probability associated with each estimate, as per the HMM belief state update equations, is split across all the duplicate nodes within the same time-step, such that the probability tied to each individual node in the tree is actually a lower bound on the true state probability. Although this violates the claim to true best-first estimate enumeration, the tree decomposition is conceptually advantageous because it facilitates the use of proven AI search techniques to efficiently determine the shortest path (most likely trajectory) to a leaf node (estimate).

As an example of the trajectory approximation, consider the same IMU system scenario that was originally shown in Figure 3-2 (Page 32) and its propagation step displayed again here in Figure 4-3a with the inconsistent states preemptively removed. Figure 4-3b shows the same scenario when using the trajectory approximation, as indicated with only a single incoming transition arc to each next state.

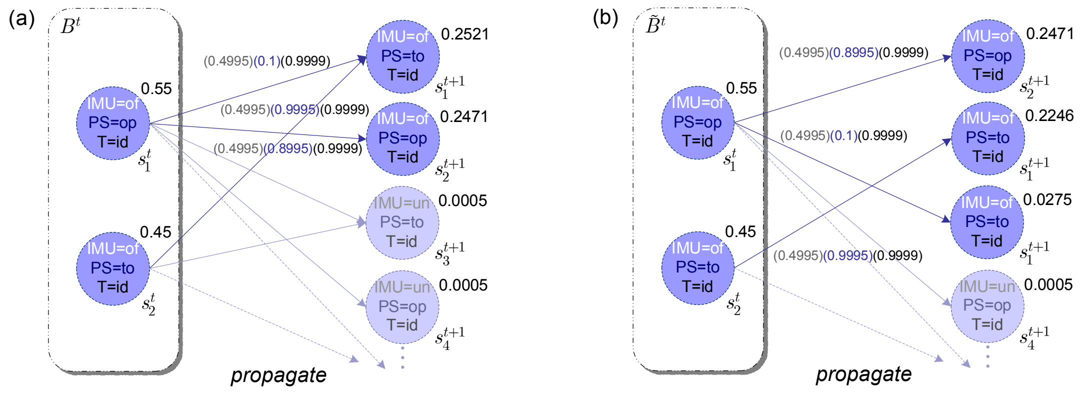

(a)
$B^t$
0.55
IMU=of
PS=op
T=id
$s_1^t$
0.45
IMU=of
PS=to
T=id
$s_2^t$
(0.4995)(0.1)(0.9999)
(0.4995)(0.9995)(0.9999)
(0.4995)(0.8995)(0.9999)
IMU=of PS=to T=id 0.2521 $s_1^{t+1}$
IMU=of PS=op T=id 0.2471 $s_2^{t+1}$
IMU=un PS=to T=id 0.0005 $s_3^{t+1}$
IMU=un PS=op T=id 0.0005 $s_4^{t+1}$
propagate
(b)
$\tilde{B}^t$
0.55
IMU=of PS=op T=id $s_1^t$
0.45
IMU=of PS=to T=id $s_2^t$
(0.4995)(0.8995)(0.9999)
(0.4995)(0.1)(0.9999)
(0.4995)(0.9995)(0.9999)
IMU=of PS=op T=id 0.2471 $s_2^{t+1}$
IMU=of PS=to T=id 0.2246 $s_1^{t+1}$
IMU=of PS=to T=id 0.0275 $s_1^{t+1}$
IMU=un PS=op T=id 0.0005 $s_4^{t+1}$
propagate

Figure 4-3: IMU Plant using (a) exact HMM propagation and (b) approximate trajectory propagation.

The three most likely trajectories for the scenario in Figure 4-3b are presented in Table 4.1, with their source state, target state, state trajectory probability, and percent error when compared to the true HMM *a priori* probabilities of Figure 4-3a. It is important to notice that since the trajectory approximation splits the two trajectories leading to $s_1^{t+1}$, this state is no longer evaluated as the most likely estimate. Even worse, the state may not be tracked at all. Incorrectly estimating the likelihood of states in this fashion can have a major affect on the action a control system will take to achieve a goal. In the case of this IMU example, the corrective action to resolve the tripped switch state will not be executed if you base your control action solely on the most likely state.

Table 4.1: Results for *Best-First Trajectory Enumeration*

| solution # | source state | target state | probability | % error |
|---|---|---|---|---|
| 1 | $s_1^t$ | $s_2^{t+1}$ | 0.2471 | 0.0 |
| 2 | $s_2^t$ | $s_1^{t+1}$ | 0.2246 | 10.9 |
| 3 | $s_1^t$ | $s_1^{t+1}$ | 0.0275 | 89.1 |

This trajectory approximation is the basis for the *Best-First Trajectory Enumeration (BFTE)*[16] estimation approach that is employed in Livingstone[21, 15], and previously in Titan [20]. Pseudo code for BFTE is provided in Appendix B. As exhibited in the example above, the trajectory approximation underestimates the true estimate probability and can lead to misdiagnosis. Section 4.3 provides the first innovative contribution of this thesis that eliminates this approximation and, furthermore, improves runtime performance.

### 4.1.3 Observation Approximation

The precise observation probability distribution was previously given in Equation 3.4, under the assumption that there is a uniform distribution across all consistent observation assignments for a given state. Computing the number of consistent observation assignments is worst case exponential in the number of model variables. Static diagnosis systems [6, 7] computed these assignments under the assumption that observations were independent. However, this can lead to significant inaccuracy due to observation coupling. To achieve real-time performance for online mode estimation, under impoverished computational resources, Livingstone made the following approximation:

$$\mathbf{P}(o^{t+1}|s_j^{t+1}) = \begin{cases} 0 & \text{if } s_j^{t+1} \wedge \mathbb{M} \models \neg o^{t+1}, \\ 1 & \text{otherwise.} \end{cases} \tag{4.1}$$

Equation 4.1 states that the observation probability is 1 when the observations $o^{t+1}$ are consistent with $s_j^{t+1} \wedge \mathbb{M}$ and 0 when they are inconsistent. Since each solution $\mathbf{x}$ to an OCSP formulation of BFTE must satisfy constraints $C(\mathbf{y})$, the update step is implicitly computed such that $\mathbf{P}(o^{t+1}|\mathbf{x})$ is 1 when observations are consistent with $C(\mathbf{y})$ and 0 otherwise. This results in an optimistic estimate that avoids searching the observation state space.

Although this approximation is computationally beneficial, its overly optimistic probability estimate can significantly deteriorate diagnostic, over both the near-term and long-term. Consider the exact single-step belief state update example for the IMU, illustrated again in Figure 4-4a. From this example, it is clear that the ordering of *a priori* estimates is directly affected by the observation probabilities, where the most likely *a priori* state $s_3^{t+1}$ gets shifted to third in the *a posteriori* ordering in $B^{t+1}$. Using Equation 4.1, the *a priori* ordering would hold (as seen in Figure 4-4b) since the observation probability is 1 when the states are consistent with the observations[2].

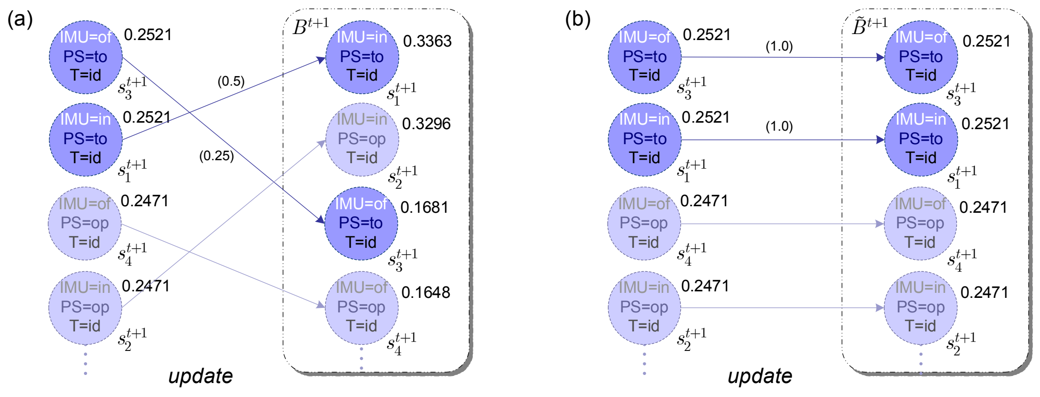

(a)
IMU=of PS=to T=id 0.2521 $s_3^{t+1}$
IMU=in PS=to T=id 0.2521 $s_1^{t+1}$
IMU=of PS=op T=id 0.2471 $s_4^{t+1}$
IMU=in PS=op T=id 0.2471 $s_2^{t+1}$
(0.5)
(0.25)
update
$B^{t+1}$
IMU=in PS=to T=id 0.3363 $s_1^{t+1}$
IMU=in PS=op T=id 0.3296 $s_2^{t+1}$
IMU=of PS=to T=id 0.1681 $s_3^{t+1}$
IMU=of PS=op T=id 0.1648 $s_4^{t+1}$
(b)
IMU=of PS=to T=id 0.2521 $s_3^{t+1}$
IMU=in PS=to T=id 0.2521 $s_1^{t+1}$
IMU=of PS=op T=id 0.2471 $s_4^{t+1}$
IMU=in PS=op T=id 0.2471 $s_2^{t+1}$
(1.0)
(1.0)
update
$\tilde{B}^{t+1}$
IMU=of PS=to T=id 0.2521 $s_3^{t+1}$
IMU=in PS=to T=id 0.2521 $s_1^{t+1}$
IMU=of PS=op T=id 0.2471 $s_4^{t+1}$
IMU=in PS=op T=id 0.2471 $s_2^{t+1}$

Figure 4-4: IMU Plant using (a) exact HMM update and (b) approximate observation probability update.

Table 4.2 lists the solutions from Figure 4-4b in order of approximate *a posteriori* probability. These results are compared with the exact probability solutions provided in Figure 4-4a and the percent error between the two is shown in the last column.

[2]Recall from Section 3.1.2 that states $s_1^{t+1}$ and $s_2^{t+1}$ in Figure 4-4 are inconsistent with $\mathbb{M}$, regardless of the observation assignments. As a result, neither state would be returned as a valid solution to the OCSP, but they are still depicted in this example to illustrate the importance of correct observation probabilites.

Table 4.2: Results for Approximate Observation Probability Update

| solution # | target state | exact probability | approximate probability | % error |
|---|---|---|---|---|
| 1 | $s_3^{t+1}$ | 0.1681 | 0.2521 | 49.9 |
| 2 | $s_1^{t+1}$ | 0.3363 | 0.2521 | 25.0 |
| 3 | $s_4^{t+1}$ | 0.1648 | 0.2471 | 49.9 |

From this example, it is clear that, not only does this observation probability approximation result in incorrect state estimate ordering, it also introduces a large amount of error in the state probabilities that are tracked over time. Section 4.4 of this chapter extends Section 4.3 to include a tractable approach to computing the correct observation probabilities and using them online during the most likely estimate search.

## 4.2 Best-First Solutions using OPSAT

Out of the three belief state update approximations that were just discussed in Section 4.1, we will continue to use Approximation 1 and eliminate Approximations 2 and 3 using novel mode estimation techniques discussed in Sections 4.3 and 4.4, respectively. This section reviews a method for efficiently generating best-first solutions to an OCSP that we will use in order to benefit from Approximation 1.

The OCSP, first discussed in Section 3.2, extends constraint satisfaction problems to optimization, by associating a utility with the assignments to decision variables. Mode estimation is framed as a specific instance of an OCSP. Although the state space of mode estimates is exponential in the number of components, Section 4.1.1 indicated that the belief state can be reasonably approximated by tracking only a handful of the most likely estimates. This section reviews OPSAT as an efficient OCSP solver that generates solutions in best-first order [22, 18], such that the most likely solutions are generated quickly.

### 4.2.1 *Conflict-directed A*$^*$

OPSAT is an OCSP solver based on *Conflict-directed A*$^*$ [22], which efficiently finds solutions to an OCSP in best-first order, by interleaving candidate generation and test.

OPSAT generates each leading candidate $\mathbf{x}$ by performing variable splitting guided by A$^*$ search, and then tests the candidate for consistency against constraints $C(\mathbf{y})$. If $\mathbf{x}$ proves inconsistent, OPSAT summarizes the inconsistency (called a *conflict*) and uses this summary to jump over other leading candidates that are similarly inconsistent. Pseudo code for the top-level loop of *Conflict-directed A$^*$* is provided in Algorithm 4.1.

**Algorithm 4.1** CONFLICT-DIRECTED A$^*$($OCSP$)

```
1: Conflicts[$OCSP$] $\leftarrow \{\}$
2: $OCSP \leftarrow$ INITIALIZE-BEST-KERNELS($OCSP$)
3: Solutions[$OCSP$] $\leftarrow \{\}$
4: loop
5:    decision-state $\leftarrow$ NEXT-BEST-STATE-RESOLVING-CONFLICTS($OCSP$)
6:    if decision-state $= \emptyset$ or TERMINATE?($OCSP$) then
7:       return Solutions[$OCSP$]
8:    if CONSISTENT?(CSP[$OCSP$], decision-state) then
9:       add decision-state to Solutions[$OCSP$]
10:   else
11:      new-conflicts $\leftarrow$ EXTRACT-CONFLICTS(CSP[$OCSP$], decision-state)
12:      Conflicts[$OCSP$] $\leftarrow$ ELIMINATE-REDUNDANT-CONFLICTS(Conflicts[$OCSP$] $\cup$
         new-conflicts)
```

After initializing, the main loop of CONFLICT-DIRECTED A$^*$ begins by calling NEXT-BEST-STATE-RESOLVING-CONFLICTS, which generates the next best valued candidate (full assignment to decision variables) that resolves all discovered conflicts. If the candidate is found to be consistent with with constraints $C(\mathbf{y})$, it is a valid solution and added to the set of solutions to the OCSP. Otherwise, EXTRACT-CONFLICTS generalizes the inconsistency into one or more conflicts by using any CSP algorithm that is capable of conflict extraction. The algorithm terminates when there is no next-best solution (the search space has been exhausted) or when TERMINATE? is true (application specific). After each loop, the new conflicts are then used by NEXT-BEST-STATE-RESOLVING-CONFLICTS (Algorithm 4.2) to avoid generating candidates are that are similarly inconsistent.

NEXT-BEST-STATE-RESOLVING-CONFLICTS determines the next best valued candidate that is consistent with Conflicts[$OCSP$] through the following two step process: The best valued kernel is first generated with NEXT-BEST-KERNEL and then expanded

**Algorithm 4.2** NEXT-BEST-STATE-RESOLVING-CONFLICTS($OCSP$)

```
1: best-kernel ← NEXT-BEST-KERNEL($OCSP$)
2: if best-kernel = $\emptyset$ then
3:     return $\emptyset$
4: else
5:     return KERNEL-BEST-STATE[$OCSP$](best-kernel)
```

into a candidate using KERNEL-BEST-STATE. A kernel is defined as a partial assignment to decision variables that resolves all known conflicts. The kernel is expanded into a candidate by choosing the best value assignments to the remaining decision variables. Both NEXT-BEST-KERNEL and KERNEL-BEST-STATE use traditional A* search to quickly find the best valued solutions, but NEXT-BEST-KERNEL searches over conflicts and KERNEL-BEST-STATE searches over all possible assignments to the remaining decision variables that were not assigned in the kernel. The difference between *Best-First Trajectory Enumeration* (used in previous approaches to mode estimation) and *Best-First Belief State Enumeration* (presented in this thesis) resides in the specification of the heuristic function that guides these A* searches [16].

### 4.2.2 Estimation using OPSAT

By solving the estimation problem as an OCSP using OPSAT, each estimate in an approximate belief state is enumerated in best-first order. As in previous mode estimation approaches, we will take advantage of OPSAT to reduce the exponential number of possible states to the $k$ most likely. The decision variables $\mathbf{x}$ are the set of mode variables $\Pi^m$ and the constraints $C(\mathbf{y})$ restrict mode variable assignments $(x_a = v'_a)$ to those that are consistent with observations $o^{t+1}$, modal constraints $\mathbb{M}_a(x_a = v'_a)$, and component interconnections $\mathbb{Q}$. For the exact PCCA estimation problem that was framed as an OCSP on Page 36, the utility function $f(\mathbf{x})$ is the HMM belief state update equations. In order to guarantee optimality and efficiently guide the NEXT-BEST-KERNEL and KERNEL-BEST-STATE A* searches of *Conflict-directed A**, an admissible heuristic with a tight optimistic bound must be specified. In the remainder of this section, we will provide a simple IMU system example using the same heuristic that was previously used in

*Best-First Trajectory Enumeration.* Section 4.3 eliminates the trajectory approximation by specifying an admissible heuristic based on the HMM propagation equation.

### 4.2.3 Example: Most Likely Trajectory for the IMU System

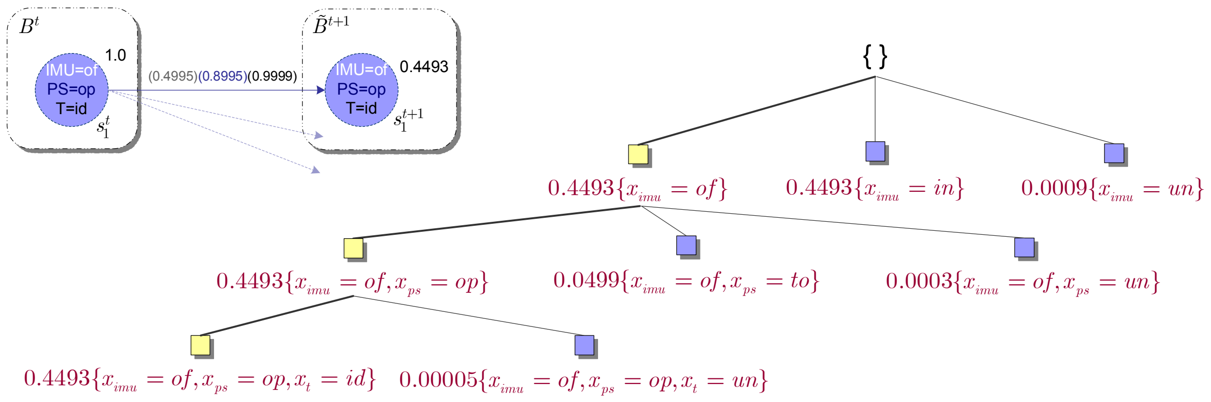
Consider the problem of calculating the most likely trajectory out of a single initial state. In this scenario, the OCSP formulation of the PCCA estimation problem is identical to that of Algorithm 3.1 except the utility function is simplified to the state trajectory equation (shown again in Equation 4.2) when there is only a single state transition into the next state. Since all the component transitions are probabilistically independent, this equation can be split into an admissible heuristic as seen in Equation 4.3.

$$f(s_j^{t+1}) = \prod_{(x_a^{t+1}=v_a')\in s_j^{t+1}} \left(\mathbf{P}(x_a^{t+1}=v_a' \mid x_a^t = v_a, s_i^t, \mu^t)\right) \cdot \mathbf{P}(s_i^t \mid o^{<0,t>}, \mu^{<0,t-1>}) \tag{4.2}$$

$$f(n) = \left( \begin{array}{ll} \prod_{(x_g^{t+1}=v_g')\in n} & \left(\mathbf{P}(x_g^{t+1}=v_g' \mid x_g^t = v_g, s_i^t, \mu^t)\right) \cdot \\ \prod_{(x_h^{t+1}=v_h')\notin n} & \max_{v_h' \in \mathbb{D}(x_h)} \left(\mathbf{P}(x_h^{t+1}=v_h' \mid x_h^t = v_h, s_i^t, \mu^t)\right) \cdot \\ & \mathbf{P}(s_i^t \mid o^{<0,t>}, \mu^{<0,t-1>}) \end{array} \right) \tag{4.3}$$

The $n$ in Equation 4.3 denotes a node in the search space and consists of a partial assignment to decision variables. The first product is the uniform-cost heuristic and represents the exact utility of the current assignments in $n$. The second product in Equation 4.3 is the greedy heuristic and represents an optimistic guess at the best value to the remaining decision variables. The heuristic is admissible because $f(n) \geq f(s_j^{t+1})$ due to maximizing each independent component transition probability that is not in $n$. *Conflict-directed A** search is guided by this heuristic in order to quickly identify the most likely trajectory.

As an example of generating the most likely trajectory using *Conflict-directed A**, consider when the IMU is *Off*, the Power Switch is *Open*, and the Timer is *Idle* with 100% certainty (upper-left of Figure 4-5). Assuming that there are no commands issued and nominal observations received, the kernel produced by NEXT-BEST-KERNEL is the empty set since there are no conflicts. KERNEL-BEST-STATE is then used to search through the most likely mode assignments using A* to find the next most likely state (Figure 4-5).



$B^t$
1.0
IMU=of
PS=op
T=id
$s_1^t$
(0.4995)(0.8995)(0.9999)
$\tilde{B}^{t+1}$
0.4493
IMU=of
PS=op
T=id
$s_1^{t+1}$
{ }
$0.4493\{x_{imu} = of\}$
$0.4493\{x_{imu} = in\}$
$0.0009\{x_{imu} = un\}$
$0.4493\{x_{imu} = of, x_{ps} = op\}$
$0.0499\{x_{imu} = of, x_{ps} = to\}$
$0.0003\{x_{imu} = of, x_{ps} = un\}$
$0.4493\{x_{imu} = of, x_{ps} = op, x_t = id\}$
$0.00005\{x_{imu} = of, x_{ps} = op, x_t = un\}$

Figure 4-5: Single Trajectory Enumeration for the IMU Plant.

In this simple example, A* searches over all the mode variables in the IMU Plant. The dark squares denote nodes that are currently in the A* queue and the light squares denote nodes that have been popped off the queue and expanded. The search begins by expanding out the root node and adding all the *reachable* IMU mode assignments to the A* priority queue. The heuristic function, defined in Equation 4.3, is used to calculate the utility of each node. For example, the uniform-cost heuristic for $\{x_{imu} = in\}$ is (0.4995) and the greedy heuristic is $(0.8995 \cdot 0.9999)$, resulting in a $(0.4995) \cdot (0.8995 \cdot 0.9999) =$ 0.4493 utility value. This is an optimistic value since an unforeseen conflict may rule out the nominal transitions of the Power Switch and Timer when the candidate is checked for consistency. The A* search continues by popping the best valued node off the queue and expanding its children in a similar manner. The final solution for this example is $\{x_{imu} = of, x_{ps} = op, x_t = id\}$ with state trajectory probability 0.4493. Generating the next-best solution is simply a matter of continuing this conflict-directed search.

This simple example has illustrated the use of OPSAT for the purpose of most likely trajectory enumeration, given a single initial state. Previous monitoring and fault diagnosis approaches have used the *Best-First Trajectory Enumeration (BFTE)* [16] technique to track multiple trajectories in an approximate belief state. BFTE specified the same search heuristic defined in Equation 4.3, but used $k$ OCSPs (one for each state in the $\tilde{B}^t$) in order to find the most likely trajectories out of each initial state, and compared them to determine the $k$ most likely trajectories overall. Pseudo code for BFTE is provided in Appendix B. The next section specifies a novel heuristic that is based on the HMM propagate equation, eliminating Approximation 2 and enabling the estimation problem to be framed as a single OCSP.

## 4.3 *Best-First Belief State Enumeration*

The previous two sections of this chapter discussed the three major approximations that have been used in mode estimation and reviewed OPSAT as an efficient way of generating estimates in best-first order. This section presents a novel mode estimation technique, call *Best-First Belief State Enumeration (BFBSE)*, that expands estimates in best-first order using a search heuristic based directly on the HMM propagate equation, eliminating Approximation 2. Although this technique still approximates the belief state by enumerating only the $k$ best estimates, BFBSE provides a more compact representation of the belief state that avoids expanding the *Trellis* structure into a branching tree, resulting in increased estimator accuracy. In addition, this compact representation enables PCCA estimation to be framed as a single OCSP, increasing the estimator runtime performance and reducing heap memory usage. Pseudo code for `BFBSE` is presented in Algorithm 4.3.

`BFBSE` is similar to Algorithm 3.1 that solved the exact PCCA estimation problem, except BFBSE only maintains the $k$ most likely estimates (Approximation 1) and leaves out the HMM update equation (Approximation 3), such that the observation probability is 1.0 when the observations are consistent with $s_j^{t+1} \wedge \mathbb{M}$ and 0.0 when they are inconsistent.

### 4.3.1 BFBSE Heuristic Function

In order for OPSAT to enumerate states in best-first order, an admissible heuristic must be specified for the OCSP utility function in `BFBSE`. Recall that the HMM propagation

**Algorithm 4.3** BFBSE($\mathcal{P}, \tilde{B}^t, \mu^t, o^{t+1}$)

```
1: Setup the OCSP $\langle \mathbf{y}, f, C \rangle$:
    • The vector $\mathbf{x}$ includes a decision variable $\mathbf{x}_a$ for each component of the plant, whose domain $\mathbb{D}(\mathbf{x}_a)$ is the set of modes that are reachable from any current state $s_i^t \in \tilde{S}^t$. For all $s_i^t \in \tilde{S}^t$, the target mode for each transition $(x_a = v_a') = \tau_a(x_a = v_a, g_a)$ whose source $(x_a = v_a) \in s_i^t$ and guard $g_a$ are satisfied by $C_M^t \wedge s_i^t \wedge \mu^t$ is considered reachable, such that $v_a' \in \mathbb{D}(\mathbf{x}_a)$. $C_M^t = \mathbb{Q} \wedge (\bigwedge_{(x_a=v_a)\in s_i^t} \mathbb{M}_a(x_a = v_a))$.
    • The utility function $f(\mathbf{x})$ is the prior probability of next state $\mathbf{x}$. More precisely, $f(\mathbf{x}) = \sum_{s_i^t \in \tilde{S}^t} \mathbf{P}(\mathbf{x} \mid s_i^t, \mu^t) \cdot p^t(s_i)$, where $\mathbf{P}(\mathbf{x} \mid s_i^t, \mu^t) = \prod_{(x_a=v_a')\in\mathbf{x}} \mathbf{P}(x_a = v_a' \mid x_a = v_a, s_i^t, \mu^t)$ and $p^t(s_i)$ is the posterior probability for state $s_i^t$.
    • $C(\mathbf{y})$ encodes the constraint that $\mathbf{x} \wedge C_{M\mathbf{x}} \wedge o^{t+1}$ must be consistent. $C_{M\mathbf{x}} = \mathbb{Q} \wedge (\bigwedge_{(x_a=v_a')\in\mathbf{x}} \mathbb{M}_a(x_a = v_a'))$.
2: Compute the $k$ most likely solutions $\tilde{S}^{t+1} = \{\mathbf{x^1}, \ldots, \mathbf{x^k}\}$ to OCSP$\langle \mathbf{y}, f, C \rangle$ in best-first order using OPSAT.
3: Extract the normalized posterior state estimate probabilities, such that $p^{t+1}(s_j) = f(s_j) / \sum_{s_i \in \tilde{S}^{t+1}} f(s_i)$ for all $k$ solutions $s_j \in \tilde{S}^{t+1}$.
4: return the $k$ most likely state estimates contained by $\tilde{B}^{t+1} = \langle \tilde{S}^{t+1}, p^{t+1} \rangle$.
```

equation (shown again in Equation 4.4) is used directly as the BFBSE utility function.

$$\mathbf{P}(s_j^{t+1} | o^{<0,t>}, \mu^{<0,t>}) = \sum_{s_i^t \in \tilde{S}^t} \left( \prod_{(x_a^{t+1}=v_a')\in s_j^{t+1}} \left(\mathbf{P}(x_a^{t+1} = v_a' \mid x_a^t = v_a, s_i^t, \mu^t)\right) \mathbf{P}(s_i^t | o^{<0,t>}, \mu^{<0,t-1>}) \right) \tag{4.4}$$

$$f(n) = \sum_{s_i^t \in \tilde{S}^t} \left( \begin{array}{ll} \prod_{(x_g^{t+1}=v_g')\in n} & \left(\mathbf{P}(x_g^{t+1} = v_g' \mid x_g^t = v_g, s_i^t, \mu^t)\right) \cdot \\ \prod_{(x_h^{t+1}=v_h')\notin n} & \max_{v_h' \in \mathbb{D}(x_h)} \left(\mathbf{P}(x_h^{t+1} = v_h' \mid x_h^t = v_h, s_i^t, \mu^t)\right) \cdot \\ & \mathbf{P}(s_i^t | o^{<0,t>}, \mu^{<0,t-1>}) \end{array} \right) \tag{4.5}$$

The BFBSE heuristic function (Equation 4.5) is derived by decomposing the HMM propagate equation into a uniform-cost heuristic and a greedy heuristic, similar to how the state trajectory equation was split in the BFTE heuristic (Equation 4.3). Leveraging on the independent component transition assumption, the state transition probabilities are broken into a uniform-cost heuristic that represents the exact utility of the current assignments in $n$, and a greedy heuristic that represents an optimistic guess at the best value to the remaining decision variables. The difference between the BFBSE heuristic and the BFTE heuristic is the summation across all incoming transitions from source states $\tilde{S}^t$ that BFBSE uses to increase the accuracy of the estimate probabilities. It is important to note that the assignment made to a particular component $x_h^{t+1}$ in the greedy heuristic might differ between incoming trajectories due to different source states $\tilde{S}^t$ that can affect the "max" computation. This is important in order to guarantee that heuristic is admissible without spending an absorbent amount of time computing the heuristic.

### 4.3.2 Example: BFBSE for the IMU Plant

As an example of how to execute `BFBSE` online, consider the same IMU Plant scenario that was originally introduced in Section 3.1.2 on Page 32. There are two initial states, $\{x_{imu} = of, x_{ps} = op, x_t = id\}$ with probability 0.55 and $\{x_{imu} = of, x_{ps} = to, x_t = id\}$ with probability 0.45, and we assume that there are no commands issued and only nominal observations received. The propagation and normalization performed by `BFBSE` when tracking 2 estimates is illustrated in Figure 4-6 with the inconsistent states preemptively removed. All the enabled transitions for this scenario are displayed again in Table 4.3 below.

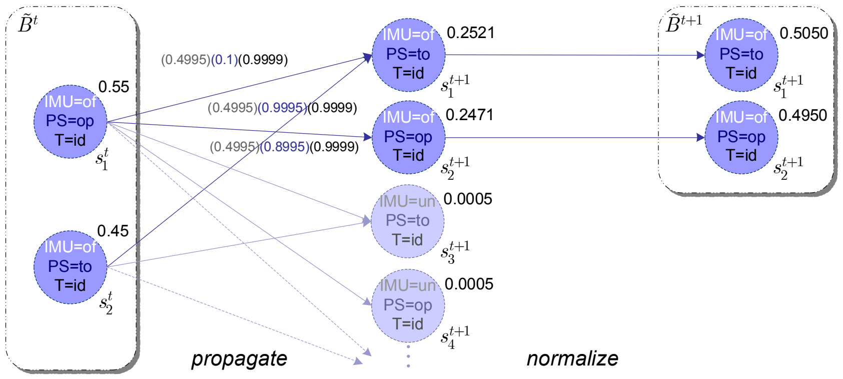


Figure 4-6: *Best-First Belief State Enumeration* for the IMU Plant.

The *Conflict-directed A** search is performed in the same way as the most likely trajectory example in Section 4.2.3. The only difference is in the BFBSE heuristic function defined in Equation 4.5. In order to generate the 2 most likely estimates in $\tilde{B}^{t+1}$, we

Table 4.3: Enabled Component Transitions for the IMU Plant, $\mathcal{P}$

| $s^t$ | $(x_{imu} = v_{imu})$ | $g_{imu}$ | $\mathbb{T}_{imu}$ | $\mathbf{P}_{\mathbb{T}_{imu}}$ |
|---|---|---|---|---|
| $s_1^t$ | $x_{imu} = of$ | (unconstrained) | $\{of, in, un\}$ | $\{0.4995, 0.4995, 0.001\}$ |
| $s_2^t$ | $x_{imu} = of$ | (unconstrained) | $\{of, in, un\}$ | $\{0.4995, 0.4995, 0.001\}$ |

| $s^t$ | $(x_{ps} = v_{ps})$ | $g_{ps}$ | $\mathbb{T}_{ps}$ | $\mathbf{P}_{\mathbb{T}_{ps}}$ |
|---|---|---|---|---|
| $s_1^t$ | $x_{ps} = op$ | $\neg(\mu_{ps}^{cmd} = close)$ | $\{op, to, un\}$ | $\{0.8995, 0.1, 0.0005\}$ |
| $s_2^t$ | $x_{ps} = to$ | $\neg(\mu_{ps}^{cmd} = open)$ | $\{to, un\}$ | $\{0.9995, 0.0005\}$ |

| $s^t$ | $(x_t = v_t)$ | $g_t$ | $\mathbb{T}_t$ | $\mathbf{P}_{\mathbb{T}_t}$ |
|---|---|---|---|---|
| $s_1^t$ | $x_t = id$ | $\neg(d_t^{im} = in)$ | $\{id, un\}$ | $\{0.9999, 0.0001\}$ |
| $s_2^t$ | $x_t = id$ | $\neg(d_t^{im} = in)$ | $\{id, un\}$ | $\{0.9999, 0.0001\}$ |

begin by expanding out the root node, as seen in Figure 4-7.

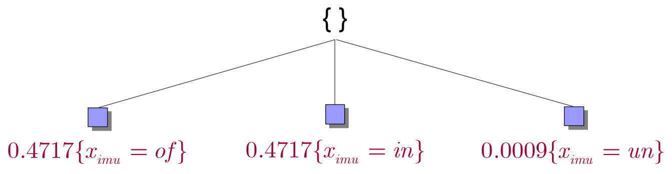


Figure 4-7: Iteration 1 BFBSE expansion for the IMU Plant.

Once again, all the *reachable* modes for the IMU are expanded and placed in the A* priority queue. The utility associated with each node represents the highest possible probability that could result when assignments are made to the remaining decision variables. Since the utility is an optimistic guess, the heuristic is admissible and guarantees that the solutions returned by A* are optimal. The heuristic calculation for the $\{x_{imu} = of\}$ node are shown below:

$$\begin{aligned} f(x_{imu} = of) &= (\mathbf{P}(x_{imu} = of|x_{imu} = of))\,(\mathbf{P}(x_{ps} = op|x_{ps} = op)\mathbf{P}(x_t = id|x_t = id))\,p(s_1^t) \\ &\quad + (\mathbf{P}(x_{imu} = of|x_{imu} = of))\,(\mathbf{P}(x_{ps} = to|x_{ps} = to)\mathbf{P}(x_t = id|x_t = id))\,p(s_2^t) \\ &= (0.4995)(0.8995 \cdot 0.9999)(0.55) + (0.4995)(0.9995 \cdot 0.9999)(0.45) \\ &= 0.4717 \end{aligned}$$

As mentioned above, the target mode assignment for any particular mode variable may not match for each incoming state transition. For the $\{x_{imu} = of\}$ node, the maximum component transition probability for the Power Switch was into the *Open* mode in the first state transition and *Tripped Open* in the second state transition. Although these two component transitions cannot combine into a single state, the heuristic allows for quick computation that is admissible and provides a tight upper bound. The heuristic is exact if all the maximum probability component transitions lead to the same target mode.

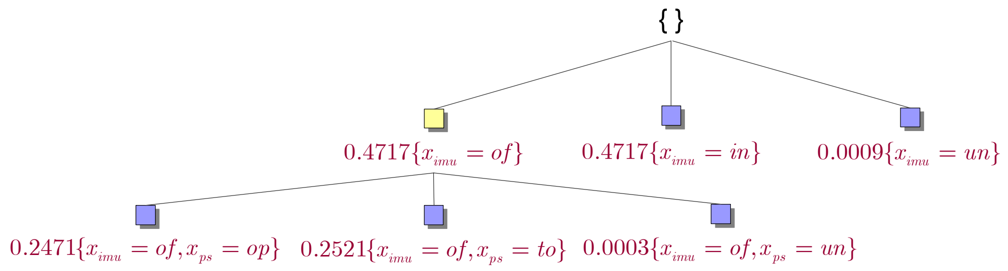

{ }
$0.4717\{x_{imu} = of\}$
$0.4717\{x_{imu} = in\}$
$0.0009\{x_{imu} = un\}$
$0.2471\{x_{imu} = of, x_{ps} = op\}$
$0.2521\{x_{imu} = of, x_{ps} = to\}$
$0.0003\{x_{imu} = of, x_{ps} = un\}$

Figure 4-8: Iteration 2 BFBSE expansion for the IMU Plant.

Iteration 2 of the BFBSE *Conflict-directed A** search, shown in Figure 4-8, begins by popping off the best value node in the queue and expands its children. In this case, $\{x_{imu} = of\}$ and $\{x_{imu} = in\}$ are equally likely so either is valid. Since the second layer of the tree assigns a value to the Power Switch, the heuristic calculation is bound to evaluating that specific transition probability and no longer takes the maximum Power Switch transition probability. The heuristic computation for the $\{x_{imu} = of, x_{ps} = to\}$ node is shown below:

$$\begin{aligned} f(x_{imu} = of, x_{ps} = to) &= (\mathbf{P}(x_{imu} = of | x_{imu} = of)\mathbf{P}(x_{ps} = to | x_{ps} = op))\,(\mathbf{P}(x_t = id | x_t = id))\,p(s_1^t) \\ &\quad + (\mathbf{P}(x_{imu} = of | x_{imu} = of)\mathbf{P}(x_{ps} = to | x_{ps} = to))\,(\mathbf{P}(x_t = id | x_t = id))\,p(s_2^t) \\ &= (0.4995 \cdot 0.1)(0.9999)(0.55) + (0.4995 \cdot 0.9995)(0.9999)(0.45) \\ &= 0.2521 \end{aligned}$$

In Iteration 3, the best cost node is $\{x_{imu} = in\}$ from the first layer in the search tree with a utility of 0.4717. This node is then expanded in a similar fashion; adding all the new children and computing their utility. Figure 4-9 illustrates this new expansion and hides the children of $\{x_{imu} = of\}$ for readability. In this scenario, the children of

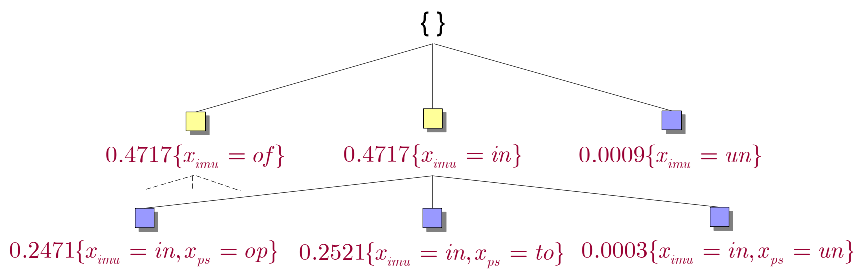

{ }
$0.4717\{x_{imu} = of\}$
$0.4717\{x_{imu} = in\}$
$0.0009\{x_{imu} = un\}$
$0.2471\{x_{imu} = in, x_{ps} = op\}$
$0.2521\{x_{imu} = in, x_{ps} = to\}$
$0.0003\{x_{imu} = in, x_{ps} = un\}$

Figure 4-9: Iteration 3 BFBSE expansion for the IMU Plant.

$\{x_{imu} = of\}$ and $\{x_{imu} = in\}$ happen to have the same values and assignments to the remaining modes. As a result, there is another tie between $\{x_{imu} = of, x_{ps} = to\}$ and $\{x_{imu} = in, x_{ps} = to\}$ in Iteration 4. We again chose to expand the furthest left node to resolve the tie.

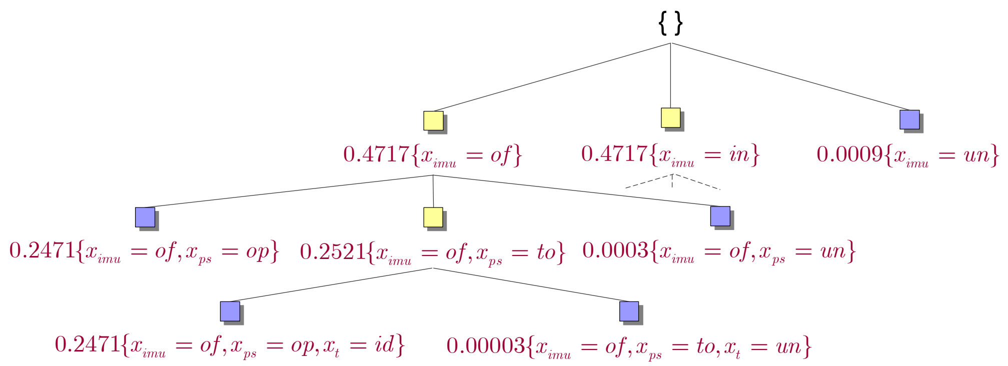

{ }
$0.4717\{x_{imu} = of\}$
$0.4717\{x_{imu} = in\}$
$0.0009\{x_{imu} = un\}$
$0.2471\{x_{imu} = of, x_{ps} = op\}$
$0.2521\{x_{imu} = of, x_{ps} = to\}$
$0.0003\{x_{imu} = of, x_{ps} = un\}$
$0.2471\{x_{imu} = of, x_{ps} = op, x_t = id\}$
$0.00003\{x_{imu} = of, x_{ps} = to, x_t = un\}$

Figure 4-10: Iteration 4 BFBSE expansion for the IMU Plant.

After Iteration 4 (Figure 4-10), both $\{x_{imu} = of, x_{ps} = to, x_t = id\}$ and $\{x_{imu} = in, x_{ps} = to\}$ nodes have the same utility of 0.2471. The far left node is again chosen and expanded in Iteration 5. At this point, $\{x_{imu} = of, x_{ps} = to, x_t = id\}$ has no more children, so it is returned by **Kernel-Best-State** as a candidate. Once **Consistent?** has determined that this candidate is in fact consistent, it is then added to the list of solutions to the OCSP. Recalling Figure 4-6, $\{x_{imu} = of, x_{ps} = to, x_t = id\}$ was correctly evaluated as the most likely consistent state in $\tilde{B}^{t+1}$. This *Conflict-directed A** search would continue until 2 solutions are found (since we are tracking 2 states) or the search space has been exhausted.

This section has introduced *Best-First Belief State Enumeration* as an approximate mode estimation technique that uses the HMM propagation equation directly as its utility function, eliminating Approximation 2. This allows for a more compact representation of the belief state evolution and increases estimator accuracy. In addition, this approach allows PCCA estimation to be framed as a single OCSP, which improves estimator run-time performance and heap memory usage. A complexity analysis and empirical results for BFBSE are provided in Chapter 5.

Although BFBSE is capable of providing increased accuracy and performance, it still relies on Approximation 3, which avoids computing the correct observation probabilities. The next section provides a tractable approach to computing the observation probabilities offline and compiling all possible combinations of observation probabilities into a compact representation that can be used online with efficient triggering techniques.

## 4.4 *Best-First Belief State Update*

This chapter began by identifying the three common assumptions that have been used by traditional monitoring and fault diagnosis approaches in order to meet the performance margins required by embedded systems. We continue to leverage Approximation 1 in order to reduce the PCCA estimation problem to the task of enumerating the $k$ most likely states in an approximate belief state, without significant loss in estimator accuracy. Section 4.3 eliminated Approximation 2 by directly using the HMM propagation equation as the OCSP utility function. This section focuses on an innovative approach to calculating the correct observation probability distribution, in order to eliminate Approximation 3. We incorporate the correct observation probabilities within BFBSE to provide a new mode estimation technique, called *Best-First Belief State Update (BFBSU)* that uses the full two-stage HMM belief state update equations as its utility function, further increasing estimator accuracy. Although this technique requires additional computation, the observation probabilities can be used to tighten the bound on the A$^*$ heuristic and provide enhanced guidance through the search space. Empirical results in Chapter 5 show that, under certain conditions, BFBSU will outperform BFBSE in time and memory.

For PCCA, a belief state is computed using the standard HMM belief state update equations. This requires *a priori* knowledge of conditional observation probabilities, as previously shown in Equation 3.2 on Page 29. The constraint-based observation probability distribution was first defined in GDE [6], and presented again here in Equation 3.4 on Page 31, but is difficult to calculate due to the large state space of sensory observations and component modes. As a result, GDE and Sherlock assumed that each observation was independent of each other, simplifying the observation probability to $\mathbf{P}(o^{t+1}|s_j^{t+1}) = \prod_{o_i \in o^{t+1}} \mathbf{P}(o_i|s_j^{t+1})$. In addition, if the single observation assignment $o_i$ was not entailed or refuted, GDE and Sherlock approximated the $1/m$ distribution by fixing the value of $m$ to $|\mathbb{D}(o_i)|$, regardless of the specific mode assignments.

Livingstone [21, 15] simplified the observation probability distribution further by assuming the observation probability is 1 or 0, depending on if the observation is simply consistent or inconsistent with the next mode assignment. For a failure mode with one or more consistent observations, the total observation probability density is incorrectly $\geq 1$ and results in a probabilistic bias toward the failure mode, eventually leading to an incorrect fault diagnosis. Recall the IMU Plant example in Figure 4-4 on Page 44. This example shows that incorrectly computing the observation probability can significantly discount the posterior state probability to the extent that the estimator no longer tracks the state as a likely hypothesis.

Our approach eliminates the observation probability approximation (Approximation 3) by counting the number of consistent observations for a candidate mode assignment, while maintaining the overall computational efficiency required for real-time mode estimation, through the following two-step process: We begin with offline generation of a compact set of observation probability rules (OPRs) that map system state to observation probabilities. Each OPR represents an entry within a conditional probability table (CPT), where the maximum size of table is the state space of the mode variables that the OPR is dependent on. During each online estimation cycle, the appropriate OPRs are quickly looked up in the CPT and used to compute the full HMM belief state update equations.

### 4.4.1 Observation Probability Rules

We first define the observation probability rule (OPR), in general, and then describe how we can leverage from the OCSP formulation and PCCA structure in order to create a compact representation of all the rules necessary to compute the precise observation probability distribution that was defined in Equation 3.4. Compactness is essential for BFBSU in order to be scalable to full-sized systems.

**Definition 4.1.** *An Observation Probability Rule is a direct mapping from a partial state assignment* $\bar{\mathbf{x}} \in \Sigma^{\mathbf{x}}$ *to the observation probability associated with the partial assignment* $\bar{\mathbf{o}} \in \Sigma^{\mathbf{o}}$ *given assignment* $\bar{\mathbf{x}}$*, such that* $\bar{\mathbf{x}} \Rightarrow \mathbf{P}(\bar{\mathbf{o}} \mid \bar{\mathbf{x}})$*. The set of partial assignments* $\Sigma^{\mathbf{x}} = \Sigma^{m}\Downarrow_{\mathbf{x}}$ *is the projection of* $\Sigma^{m}$ *onto* $\mathbf{x} \subseteq \Pi^{m}$ *and* $\Sigma^{\mathbf{o}} = \Sigma^{o}\Downarrow_{\mathbf{o}}$ *is the projection of* $\Sigma^{o}$ *onto* $\mathbf{o} \subseteq \Pi^{o}$*.*

Each OPR states that for a partial assignment to mode variables $\bar{\mathbf{x}}$, there is a specific observation probability $\mathbf{P}(\bar{\mathbf{o}} \mid \bar{\mathbf{x}})$, regardless of the actual partial observation assignment $\bar{\mathbf{o}}$. As an example, one OPR for the IMU Plant is $\{x_{imu} = of\} \Rightarrow 1/2$ since the data valid flag $o_{imu}^{dv}$ can be true or false when the IMU is *Off*. There are two assumptions that make this claim legitimate: Given the PCCA observation probability distribution defined in Equation 3.4, and displayed again here in Equation 4.6, there is a uniform probability distribution across all observations that are consistent with $s_j^{t+1} \wedge \mathbb{M}$. In other words, the observation probability is the same regardless of the particular observation assignment. In the case where the observation is inconsistent with $s_j^{t+1} \wedge \mathbb{M}$, the probability is zero and the uniform generalization does not hold. Fortunately, since we have framed estimation as an OCSP, OPSAT will automatically determine the candidate to be inconsistent and discard it. Due to the uniform probability distribution across all consistent observations and OPSAT discarding inconsistent assignments, the precise observation probability can be specified using OPRs.

$$\mathbf{P}(o^{t+1}|s_j^{t+1}) = \begin{cases} 1 & \text{if } s_j^{t+1} \wedge \mathbb{M} \models o^{t+1}, \\ 0 & \text{if } s_j^{t+1} \wedge \mathbb{M} \models \neg o^{t+1}, \\ 1/m & \text{otherwise,} \end{cases} \tag{4.6}$$

where $m$ = number of consistent assignments to $o^{t+1}$ for $s_j^{t+1}$ and $\mathbb{M}$.

One approach to using OPRs is to have a single rule for each combination of full assignments to mode variables $\bar{\mathbf{x}} \in \Sigma^m$. In this brute force approach, the number of OPRs is exponential in the number of components and is obviously intractable for large systems. More precisely, the maximum number of OPRs is $\prod_{x_a \in \Pi^m} |\mathbb{D}(x_a)|$. This problem is exacerbated by the NP-hard task of determining how many consistent observations there are for a given state when calculating the probability associated with each rule.

Fortunately, the number of OPRs can be greatly reduced by leveraging our OCSP formulation as well as the sparse interconnections between different modes and observations. The simple reduction comes from recognising that the candidate solutions to the OCSP used in BFTE and BFBSE have an intrinsic observation probability of 0 or 1, depending on if the candidate is inconsistent or consistent, respectively. For example, if a candidate is found to be inconsistent, it is removed from the list of possible solutions. This is equivalent to assigning an observation probability of zero. Likewise, if the candidate is consistent, its utility value remains unchanged as if applying an observation probability of one. Since the 0 and 1 probability values are already provided in OCSP solutions, any OPRs that map to a probability of 0 or 1 are superfluous and can be deleted. A more substantial reduction in the number of OPRs comes from a divide-and-conquer approach that decomposes the OPR state space, by identifying which observation variables are dependent on which components. This is done by calculating a dependency hypergraph between observation variables and mode variables. Since the majority of sensory observations are only dependent on a small subset of all possible components, the number of OPRs is greatly reduced. For example, if all the observation variables were only dependent on $x_a$, the precise number of OPRs is reduced to $|\mathbb{D}(x_a)|$. Using these two reduction techniques, the maximum number of OPRs for the IMU Plant is reduced from $5 \cdot 4 \cdot 4 = 80$ to the 4 shown below. In a larger model of a Mars Entry Decent and Landing (EDL) subsystem [13], with 10 mode variables, the number of OPRs is reduced from $1,458,000$ to 307. The use of the OPRs during online best-first estimate search is described in Section 4.4.3.

$$\{x_{imu} = of\} \Rightarrow 1/2, \quad \{x_{imu} = un\} \Rightarrow 1/2,$$
$$\{x_t = id\} \Rightarrow 1/2, \quad \{x_t = un\} \Rightarrow 1/2.$$

### 4.4.2 Offline Generation of Observation Probability Rules

Now that we have defined an OPR and provided some intuition on how the set of OPRs can be compactly represented, this section describes how the OPRs are generated offline. This process includes enumerating all relevant rules within a conditional probability table and computing their precise observation probability, by counting the number of consistent observation assignments.

The observation probability rules are quickly identified from a set of dissents (diagnosis rules) that are generated during offline model compilation [8, 4]. A dissent is a mapping from a partial assignment to observation variables to a conflict. Given a set of observations, the primary purpose of dissents is to quickly identify all conflicts through rule triggering, *before* performing *Conflict-directed A*$^*$ search, instead of "discovering" conflicts online using an exponential satisfiability engine.

**Definition 4.2.** *A dissent is a mapping from a minimal partial assignment to observation variables* $\bar{\mathbf{o}}$ *to a conflict* $\bar{\mathbf{x}}$*, such that* $\bar{\mathbf{o}} \Rightarrow \neg(\bar{\mathbf{x}})$*. A conflict* $\bar{\mathbf{x}}$ *is a minimal partial assignment to mode variables that is inconsistent with* $\bar{\mathbf{o}} \wedge \mathbb{M}(\bar{\mathbf{x}}) \wedge \mathbb{Q}$*.*

Intuitively, the dissent declares that if observations $\bar{\mathbf{o}}$ have been received, $\bar{\mathbf{x}}$ is inconsistent and cannot be true. Dissents are useful for the purpose of generating OPRs, since they implicitly specify which combinations of observations are inconsistent with which mode variable assignments. We determine the compact set of OPRs as well as the observation probability associated with each rule using the following four step process: (1) We begin by constructing a hypergraph based on the dependencies between observation variables and mode variables, using all the dissents in the compiled model. We then separate the hypergraph into maximally connected subgraphs. (2) Conditional probability tables (CPTs) are created for each subset of mode variables $\mathbf{x}$ contained by the subgraphs, where each element of the CPT is an OPR. (3) We then compute the maximum number of consistent observation assignments for each OPR by simply calculating the state space of $\mathbf{o}$, and subtract the maximum number of consistent observations by the number of inconsistent observations, using the dissents. The uniform $1/m$ conditional observation probability $\mathbf{P}(\bar{\mathbf{o}} \mid \bar{\mathbf{x}})$ is the inverse of the remaining number of consistent observations $m$. Finally, (4) we remove all the OPRs in the conditional probability table that have a probability of 0 or 1. The top-level pseudo code is provided in Algorithm 4.4.

**Algorithm 4.4** GENERATE-OPRS($dissents$)

```
1: $dh \leftarrow$ CREATE-OPR-DEPENDENCY-HYPERGRAPH($dissents$)
2: $cpts \leftarrow$ EXTRACT-CPTS-FROM-DEPENDENCY-HYPERGRAPH($dh$)
3: for all $opr \in cpts$ do
4:     $max\text{-}num\text{-}consistent \leftarrow$ COMPUTE-MAX-CONSISTENT($opr$)
5:     $num\text{-}consistent \leftarrow max\text{-}num\text{-}consistent -$ NUM-INCONSISTENT($opr, dissents$)
6:     $opr$ probability $\leftarrow 1/num\text{-}consistent$
7:     if $opr$ probability $= 0$ or $opr$ probability $= 1$ then
8:         remove $opr$ from $cpts$
9: return oprs
```

CREATE-OPR-DEPENDENCY-HYPERGRAPH computes a dependency hypergraph by placing virtual edges between each observation and mode variable $\mathbf{o} \cup \mathbf{x}$ in each dissent. This connects together all observation and mode variables that are dependent on one another. As an example, consider all the relevant dissents for the IMU plant shown below[3]:

$$(o_{imu}^{dv} = true) \Rightarrow \neg(x_{imu} = in) \qquad (o_t^{al} = tripped) \Rightarrow \neg(x_t = ru)$$

$$(o_{imu}^{dv} = true) \Rightarrow \neg(x_{imu} = si) \qquad (o_t^{al} = not\text{-}tripped) \Rightarrow \neg(x_t = ex)$$

$$(o_{imu}^{dv} = false) \Rightarrow \neg(x_{imu} = me)$$

[3]Dissents are only relevant to OPR generation if $\mathbf{o^*} \neq \{\}$. In the case where $\mathbf{o^*} = \{\}$, the dissent indicates mutually inconsistent mode assignments but provides no information about observation probabilities. The complete list of dissents for the compiled IMU Plant are provided in Appendix A.3.

Starting with this set of dissents, we use CREATE-OPR-DEPENDENCY-HYPERGRAPH to compute a dependency hypergraph by iterating through each dissent, placing edges between each observation and mode variables in $\mathbf{o} \cup \mathbf{x}$ across all dissents. In this simple example, there are only two hyperedges that link $x_{imu}$ to $o^{dv}_{imu}$ and $x_t$ to $o^{al}_t$, as shown in Figure 4-11. The subgraph on the left of Figure 4-11 was created by linking the observations variables and mode variables of the three dissents listed above on the left. The right subgraph was generated using the other two dissents above.

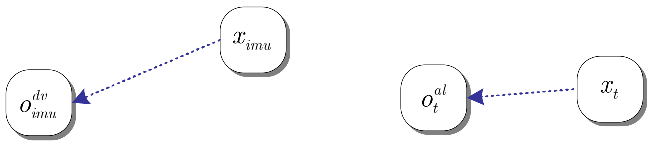


Figure 4-11: Observation probability rule dependency hypergraph for the IMU Plant.

Based on this dependency hypergraph, we can create a set of CPTs using EXTRACT-CPTS-FROM-DEPENDENCY-HYPERGRAPH. In general, this task consists of taking the cross product of all assignments to mode variables in each maximally connected subgraph. The OPRs in each CPT are used compute the observation probability for the subset of observation variables that were connected in the subgraph. Since the OPR antecedents in each CPT are mutually exclusive, only one OPR from each CPT can be triggered at once. In this example, there are two maximally connected subgraphs and each subgraph only has one mode variable. Thus, the number of OPRs is just $|\mathbb{D}(x_{imu})| + |\mathbb{D}(x_t)| = 9$ since there are two separate CPTs. In the worst case, all of the observation variables would be dependent on all of the mode variables, resulting in one CPT of size $\prod_{x_a \in \Pi^m} |\mathbb{D}(x_a)|$.

Recall that the maximum number of consistent observation assignments for each OPR is $\Sigma^{\mathbf{o}}$, where $\mathbf{o}$ is the set of all observation variables for the CPT that contains the OPR. More precisely, COMPUTE-MAX-CONSISTENT will return $\prod_{o_i \in \mathbf{o}} |\mathbb{D}(o_i)|$. In this example, the maximum number of consistent observation assignments is 2 for all OPRs, since the domain size of $o^{dv}_{imu}$ and $o^{al}_t$ are both 2.

We compute NUM-INCONSISTENT, by counting the inconsistent observation assignments provided by the dissents. The easiest approach to counting the number of inconsistent observations for each OPR is to start with a list that enumerates all observation assignments $L = \Sigma^{\mathbf{o}_{\mathcal{O}}}$, where $\mathbf{o}_{\mathcal{O}}$ is the set of all observation variables that the OPR is computing the probability for, and assume that they are all consistent. For each *relevant* dissent we mark each element in $L$ as being inconsistent, if $\mathbf{o}_{\mathcal{D}} \subseteq \mathbf{o}_{\mathcal{O}}$. A *relevant* dissent contains the same mode variables $\mathbf{x}_{\mathcal{O}}$, as the antecedent of the OPR, such that $\mathbf{x}_{\mathcal{D}} \subseteq \mathbf{x}_{\mathcal{O}}$, where $\mathbf{x}_{\mathcal{D}}$ denotes the conflict variables in the dissent. We then simply count all the elements in $L$ marked inconsistent and subtract it from the maximum number of consistent observations. For this simple example, all we do is subtract 1 inconsistent observation assignment from the maximum of 2 consistent assignments for each OPR that corresponds to an IMU Plant dissent. The inverse of the number of consistent observations is the observation probability. The results are shown below:

$$\begin{array}{rl}
(x_{imu} = of) \Rightarrow \frac{1}{2} & (x_t = id) \Rightarrow \frac{1}{2} \\
(x_{imu} = in) \Rightarrow 1 & (x_t = ru) \Rightarrow 1 \\
(x_{imu} = me) \Rightarrow 1 & (x_t = ex) \Rightarrow 1 \\
(x_{imu} = si) \Rightarrow 1 & (x_t = un) \Rightarrow \frac{1}{2} \\
(x_{imu} = un) \Rightarrow \frac{1}{2} &
\end{array}$$

Recalling that the OPRs that result in an observation probability of 0 or 1 are superfluous, the list of essential OPRs can be reduced to the following 4:

$$\begin{array}{rl}
(x_{imu} = of) \Rightarrow \frac{1}{2} & (x_t = id) \Rightarrow \frac{1}{2} \\
(x_{imu} = un) \Rightarrow \frac{1}{2} & (x_t = un) \Rightarrow \frac{1}{2}
\end{array}$$

Since the observation probabilities in each CPT is independent of the observations in other CPTs and each OPR in a CPT is mutually exclusive, the total observation probability is the product of the triggered rules. These rules are triggered online when the OPR implicant is entailed. For example, the observation probability for $\{x_{imu} = of, x_{ps} = op, x_t = id\}$ is $\frac{1}{2} \cdot \frac{1}{2} = \frac{1}{4}$.

There are other solutions to computing the number of consistent observations that are more elegant and require less offline computation. One such method is to use the observation assignments, for each *relevant* dissent, as a constituent kernel, and solve for the kernels by computing the minimal set covering. This process is similar to candidate generation introduced in GDE [6]. From the kernels, it is easy to compute all the extensions as all possible inconsistent observations. A second approach is to take the same inconsistent observation assignments from the *relevant* dissents and place them into a Binary Decision Diagram (BDD) [2]. Using the BDD formulation, it is easy to compute the total number of inconsistent observations by calling *Satisfy-count* within the BDD package. This operation has a time complexity of $O(|G|)$, where $|G|$ is the number of vertices in the BDD. The number of vertices is worst case exponential.

### 4.4.3 Online BFBSU using Observation Probability Rules

Although computing the observation probability distribution is exponential in the size of the largest hypergraph component, we retain real-time performance by shifting this computation offline. This reduces the exponential satisfiability computation to the linear process of online rule triggering [4]. Accuracy of online mode estimation is increased by extending BFBSE of Section 4.3 to efficiently compute the estimate probabilities directly from the complete HMM belief state update equations during its conflict-directed search. Pseudo code for this novel mode estimation technique, called *Best-First Belief State Update (BFBSU)*, is provided in Algorithm 4.5.

It is important to note that the BFBSU OCSP formulation is nearly identical to the exact PCCA estimation formulation on Page 36, except that we only track the $k$ most likely estimates in an approximate belief state $\tilde{B}$. This formulation has eliminated both Approximations 2 (using the HMM propagation equation) and 3 (using the HMM update equation), described in Section 4.1, to provide a highly accurate approximate mode estimation capability. BFBSU is an improvement over BFBSE because, in addition to using the HMM propagation equation (Equation 3.1), BFBSU also folds in the HMM update equation (Equation 3.2) directly into its utility function. A tractable approach

**Algorithm 4.5** BFBSU($\mathcal{P}, \tilde{B}^t, \mu^t, o^{t+1}$)

```
1: Setup the OCSP $\langle \mathbf{y}, f, C \rangle$:
   • The vector $\mathbf{x}$ includes a decision variable $\mathbf{x}_a$ for each component of the plant, whose domain $\mathbb{D}(\mathbf{x}_a)$ is the set of modes that are reachable from any current state $s_i^t \in \tilde{S}^t$. For all $s_i^t \in \tilde{S}^t$, the target mode for each transition $(x_a = v'_a) = \tau_a(x_a = v_a, g_a)$ whose source $(x_a = v_a) \in s_i^t$ and guard $g_a$ are satisfied by $C_M^t \wedge s_i^t \wedge \mu^t$ is considered reachable, such that $v'_a \in \mathbb{D}(\mathbf{x}_a)$. $C_M^t = \mathbb{Q} \wedge (\bigwedge_{(x_a=v_a)\in s_i^t} \mathbb{M}_a(x_a = v_a))$.
   • The utility function $f(\mathbf{x})$ is the posterior probability of next state $\mathbf{x}$. More precisely, $f(\mathbf{x}) = \left(\sum_{s_i^t \in S^t} \mathbf{P}(\mathbf{x} \mid s_i^t, \mu^t) \cdot p^t(s_i)\right) \cdot \mathbf{P}(o^{t+1} \mid \mathbf{x})$, where $\mathbf{P}(\mathbf{x} \mid s_i^t, \mu^t) = \prod_{(x_a=v'_a)\in \mathbf{x}} \mathbf{P}(x_a = v'_a \mid x_a = v_a, s_i^t, \mu^t)$, $p^t(s_i)$ is the posterior probability for state $s_i^t$, and $\mathbf{P}(o^{t+1} \mid \mathbf{x})$ is the observation probability for $\mathbf{x}$.
   • $C(\mathbf{y})$ encodes the constraint that $\mathbf{x} \wedge C_{M_\mathbf{X}} \wedge o^{t+1}$ must be consistent. $C_{M_\mathbf{X}} = \mathbb{Q} \wedge (\bigwedge_{(x_a=v'_a)\in \mathbf{x}} \mathbb{M}_a(x_a = v'_a))$.
2: Compute the $k$ most likely solutions $\tilde{S}^{t+1} = \{\mathbf{x^1}, \ldots, \mathbf{x^k}\}$ to OCSP$\langle \mathbf{y}, f, C \rangle$ in best-first order using OPSAT.
3: Extract the normalized posterior state estimate probabilities, such that $p^{t+1}(s_j) = f(s_j) / \sum_{s_i \in \tilde{S}^{t+1}} f(s_i)$ for all $k$ solutions $s_j \in \tilde{S}^{t+1}$.
4: **return** the $k$ most likely state estimates contained by $\tilde{B}^{t+1} = \langle \tilde{S}^{t+1}, p^{t+1} \rangle$.
```

to generating a compact set of observation probability rules was presented earlier in this section in order to provide fast online heuristic computation. The heuristic function for the BFBSU *Conflict-directed A** search is provided in Equation 4.8 below:

$$\mathbf{P}(s_j^{t+1} | o^{<0,t+1>}, \mu^{<0,t>}) = \sum_{s_i^t \in \tilde{S}^t} \left( \prod_{(x_a^{t+1}=v'_a)\in s_j^{t+1}} \left(\mathbf{P}(x_a^{t+1} = v'_a \mid x_a^t = v_a, s_i^t, \mu^t)\right) \cdot \mathbf{P}(s_i^t | o^{<0,t>}, \mu^{<0,t-1>}) \right) \mathbf{P}(o^{t+1} \mid s_j^{t+1}) \tag{4.7}$$

$$f(n) = \sum_{s_i^t \in \tilde{S}^t} \left( \prod_{(x_g^{t+1}=v'_g)\in n} \left(\mathbf{P}(x_g^{t+1} = v'_g \mid x_g^t = v_g, s_i^t, \mu^t)\right) \cdot \prod_{(x_h^{t+1}=v'_h)\notin n} \max_{v'_h \in \mathbb{D}(x_h)} \left(\mathbf{P}(x_h^{t+1} = v'_h \mid x_h^t = v_h, s_i^t, \mu^t)\right) \cdot \mathbf{P}(s_i^t | o^{<0,t>}, \mu^{<0,t-1>}) \right) \mathbf{P}(o^{t+1} \mid n) \tag{4.8}$$

Equation 4.7 is the combined form of the HMM belief state update equations without the normalization factor in the update step. The BFBSU heuristic function (Equation 4.8) is the same HMM belief state update equation, but factored into a uniform-cost heuristic and a greedy heuristic in the same way BFBSE was factored.

The innovation in BFBSU is the addition of the observation probability as part of the OCSP utility function, as well as the heuristic function that will help guide the A$^*$ search towards the most likely estimate. As A$^*$ explores deeper into the search tree, more mode assignments will be made and will trigger an increasing number of OPRs. These observation probabilities will tighten the heuristic value for the mode assignment as the search gets closer to finding a full candidate assignment. A tight bound on the heuristic function is important because it prevents the search from enumerating highly sub-optimal assignments. $\mathbf{P}(o^{t+1} \mid n)$ represents an optimistic estimate of the observation probability for the partial assignment to mode variables $n$. Since the observation probability is 1 until an OPR specifies otherwise, $\mathbf{P}(o^{t+1} \mid n) \geq \mathbf{P}(o^{t+1} \mid c^*)$, where $c^*$ is the optimal extension to $n$. Hence, the new heuristic function is admissible.

## 4.4.4 Example: BFBSU for the IMU Plant

BFBSU is demonstrated using the same IMU Plant scenario that was originally introduced in Figure 3-2 on Page 32 and also used in the BFBSE example in Section 4.3.2 of this chapter. This scenario is illustrated again in Figure 4-12 with the update step separated into an observation probability update, followed by normalization, as specified by the OCSP used in `BFBSU`.

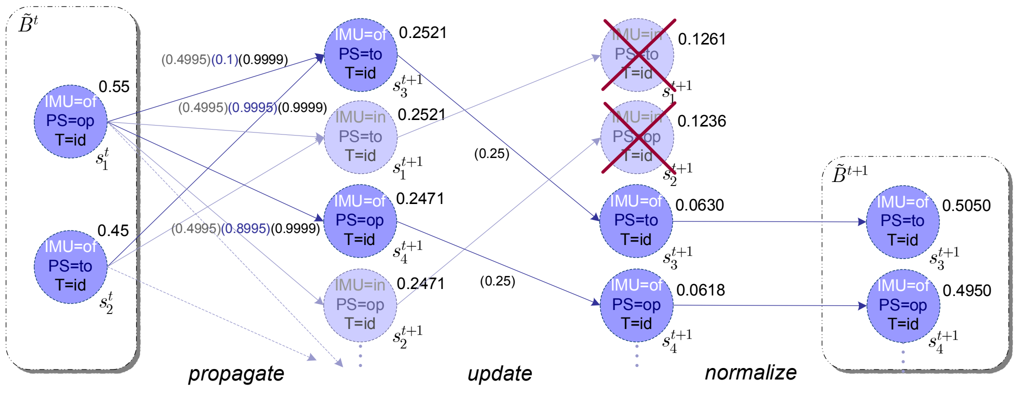


Figure 4-12: *Best-First Belief State Update* for the IMU Plant.

The best-first *Conflict-directed A$^*$* search of BFBSU is nearly identical to that of

BFBSE, except for the additional observation probability in the heuristic function. Figure 4-13 begins the search by expanding the root node, calculating the heuristic values to all of the root's children, and placing them onto the A* priority queue.

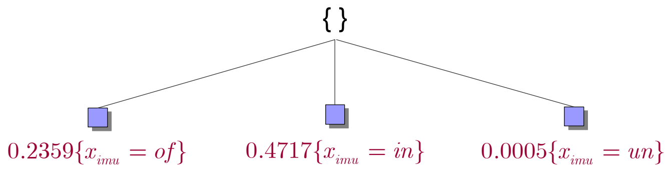


Figure 4-13: Iteration 1 BFBSU expansion for the IMU Plant.

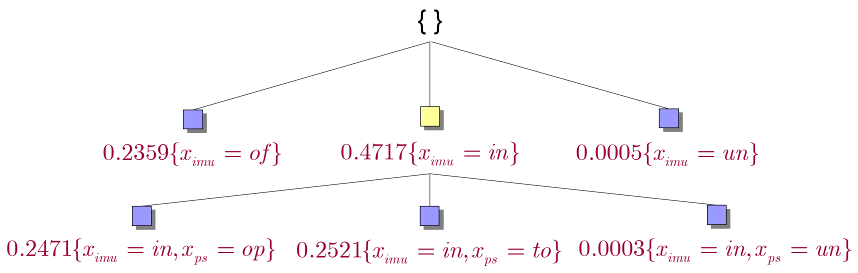
Notice that the utility values for the Layer 1 nodes are significantly different from the utility values computed for Layer 1 of the BFBSE search (Figure 4-7). Recall from the previous section that there were two OPRs for the IMU mode variable: $(x_{imu} = of) \Rightarrow \frac{1}{2}$ and $(x_{imu} = un) \Rightarrow \frac{1}{2}$. Since the node $\{x_{imu} = of\}$ assigns the *Off* value to the IMU mode, the $(x_{imu} = of) \Rightarrow \frac{1}{2}$ rule is entailed and fired. The precise heuristic calculation is shown below:

$$\begin{aligned} f(x_{imu} = of) &= \left( \begin{array}{l} (\mathbf{P}(x_{imu} = of | x_{imu} = of))\,(\mathbf{P}(x_{ps} = op | x_{ps} = op)\mathbf{P}(x_t = id | x_t = id))\,p(s_1^t) \\ +\ (\mathbf{P}(x_{imu} = of | x_{imu} = of))\,(\mathbf{P}(x_{ps} = to | x_{ps} = to)\mathbf{P}(x_t = id | x_t = id))\,p(s_2^t) \end{array} \right) \\ &\quad \cdot\ \mathbf{P}(o^{t+1} \mid x_{imu} = of) \\ &= ((0.4995)(0.8995 \cdot 0.9999)(0.55) + (0.4995)(0.9995 \cdot 0.9999)(0.45)) \cdot 0.5 \\ &= 0.2359 \end{aligned}$$

The node $\{x_{imu} = in\}$, on the other hand, does not have any OPRs associated with it. As a result, an optimistic value of 1.0 is given to the observation probability as seen below. BFBSU is innovative in the way that it uses the observation probabilities to efficiently guide the search towards the best solution.

$$\begin{aligned} f(x_{imu} = in) &= \left( \begin{array}{l} (\mathbf{P}(x_{imu} = in | x_{imu} = of))\,(\mathbf{P}(x_{ps} = op | x_{ps} = op)\mathbf{P}(x_t = id | x_t = id))\,p(s_1^t) \\ +\ (\mathbf{P}(x_{imu} = in | x_{imu} = of))\,(\mathbf{P}(x_{ps} = to | x_{ps} = to)\mathbf{P}(x_t = id | x_t = id))\,p(s_2^t) \end{array} \right) \\ &\quad \cdot\ \mathbf{P}(o^{t+1} \mid x_{imu} = in) \\ &= ((0.4995)(0.8995 \cdot 0.9999)(0.55) + (0.4995)(0.9995 \cdot 0.9999)(0.45)) \cdot 1.0 \\ &= 0.4717 \end{aligned}$$

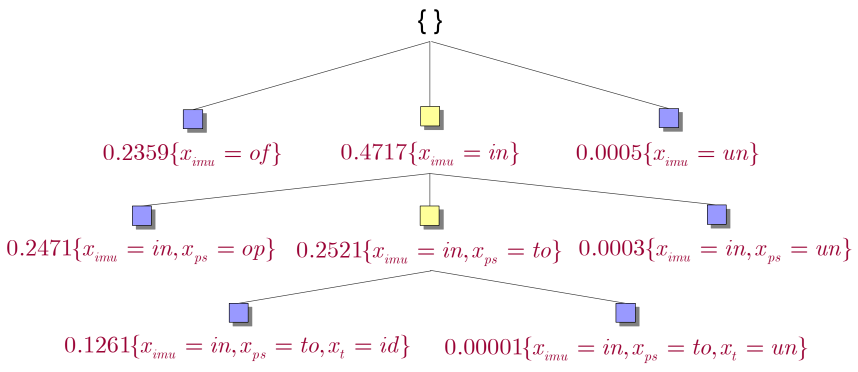
Now, the best valued node in the priority queue is clearly $\{x_{imu} = in\}$. This node is popped off the queue and expanded in Figure 4-14.



{}
$0.2359\{x_{imu} = of\}$
$0.4717\{x_{imu} = in\}$
$0.0005\{x_{imu} = un\}$
$0.2471\{x_{imu} = in, x_{ps} = op\}$
$0.2521\{x_{imu} = in, x_{ps} = to\}$
$0.0003\{x_{imu} = in, x_{ps} = un\}$

Figure 4-14: Iteration 2 BFBSU expansion for the IMU Plant.

In Layer 2 of Figure 4-14, none of the nodes have any applicable OPRs so we maintain the optimistic observation probability of 1.0. As a result, the heuristic values for this layer are identical to those of BFBSE.



{}
$0.2359\{x_{imu} = of\}$
$0.4717\{x_{imu} = in\}$
$0.0005\{x_{imu} = un\}$
$0.2471\{x_{imu} = in, x_{ps} = op\}$
$0.2521\{x_{imu} = in, x_{ps} = to\}$
$0.0003\{x_{imu} = in, x_{ps} = un\}$
$0.1261\{x_{imu} = in, x_{ps} = to, x_t = id\}$
$0.00001\{x_{imu} = in, x_{ps} = to, x_t = un\}$

Figure 4-15: Iteration 3 BFBSU expansion for the IMU Plant.

Both nodes in Layer 3 of Figure 4-15 have OPRs associated with them since it is consistent for the alarm observation of the timer to be *Tripped* or *Not Tripped* in both the *Idle* and *Unknown* modes. As a result, the uniform observation probability of $\frac{1}{2}$ is applied to both nodes.

The search continues along the same fashion until a candidate (full assignment to mode variables) is popped off the queue. In this scenario, the candidate with the best utility is $\{x_{imu} = in, x_{ps} = to, x_t = id\}$. After the candidate is generated, *Conflict-directed A** tests the candidate for consistency and determines that the candidate is inconsistent and not a valid solution. This inconsistency is generalized into the conflict $\neg(x_{imu} = in \wedge x_{ps} = to)$, which prunes the search space to avoid expanding any nodes that are extensions to that conflict. The resulting two most likely estimates using BFBSU were shown in Figure 4-12.

This chapter has discussed approximate techniques for PCCA estimation as a tractable solution to the exact PCCA estimation problem. Previous approaches to monitoring and fault diagnosis made three key assumptions in order to achieve performance demands require by embedded systems, while attempting to preserve estimator accuracy. We have presented BFBSE and BFBSU as improvements to PCCA estimation accuracy by eliminating 2 of the 3 major assumptions. BFBSE computes an approximate belief state using the HMM propagation equation directly as its utility function, and BFBSU extends the BFBSE utility function to use the full two-stage HMM belief state update equations, by generating a compact set of observation probability rules that can be efficiently triggered online. In Chapter 5, we will compare the accuracy and performance of previous estimation techniques to BFBSE and BFBSU, through theoretical analysis and empirical results.

# Chapter 5

# Results and Discussion

This thesis has focused on two novel approximate mode estimation techniques, *Best-First Belief State Enumeration (BFBSE)* and *Best-First Belief State Update (BFBSU)*, that achieve greater estimate accuracy through direct use of the HMM belief state update equations. These improvements in estimate accuracy were indicated in Chapter 4 by the elimination of 2 (out of 3) significant assumptions that were previously used in order to meet stringent performance requirements set by severely constrained on-board flight computers. In order for BFBSE an BFBSU to be ubiquitous across a wide range of embedded systems, they must also meet these strict requirements and be scalable to increasingly complex systems. This chapter presents theoretical and empirical results that show increased estimator accuracy as well as improved performance, through a reduction in memory usage and computation time, when compared to previous mode estimation techniques.

## 5.1 Space and Time Complexity

As discussed in Chapter 1, previous approaches to mode estimation, including GDE/Sherlock [6, 7], GDE+ [19], Livingstone [21, 15], model-checking [5] and Titan Mode Estimation [20], used a technique called *Best-First Trajectory Enumeration (BFTE)* to approximate the estimates by tracking a set of state trajectories and computing the estimate probabilities using a variant of the Viterbi algorithm. BFTE uses the same OCSP formulation as BFBSE and BFBSU, but differs by solving $k$ OCSP problems (one for each source state) to determine the $k$ most likely single-step trajectories out of the current approximate belief state. A description of BFTE, along with pseudo code, is provided in Appendix B.

BFTE, BFBSE, and BFBSU all frame the PCCA estimation problem as instances of an OCSP that can be solved efficiently using the Conflict-directed A$^*$ algorithm employed in OPSAT. Since the OCSP constraints are identical in BFTE, BFBSE, and BFBSU; every technique will identify the same conflicts for a given candidate solution. This allows us to compare the algorithms by evaluating their space and time complexities based on the A$^*$ candidate search portion of OPSAT.

There are two fundamental reasons why the performance of the BFBSE and BFBSU differ from the BFTE: BFBSE and BFBSU generate estimates using only one instance of OPSAT, instead of $k$ instances as performed in BFTE; and each node generated by BFBSE and BFBSU requires $k$ times more arithmetic computations than BFTE since we are summing over $k$ incoming state transitions. BFBSU differs from BFBSE by the additional observation probability lookup that is required to compute the HMM update. To more clearly understand the complexity analysis, recall that the best case time and space for A$^*$ is roughly $n \cdot b$ and the worst case time and space is $b^n$, where $b$ is the branching factor and $n$ is the depth of the tree. For our OCSP formulation, $b$ is the average number of reachable modes per component $|\mathbb{D}(x_a)|$ and $n$ is the number of components in the model $|\Pi^m|$. Table 5.1 shows the complexities for BFTE, BFBSE, and BFBSU, as an augmented form of A$^*$ search.

Table 5.1: Space and Time Complexity for BFTE, BFBSE, and BFBSU

| | Best Case | | Worst Case | |
|---|---|---|---|---|
| | Space | Time | Space | Time |
| BFTE | $k \cdot n$ | $k \cdot n \cdot (n + C)$ | $k \cdot b^n$ | $k \cdot b^n \cdot (n + C)$ |
| BFBSE | $n \cdot b$ | $n \cdot b \cdot (n \cdot k + C)$ | $b^n$ | $b^n \cdot (n \cdot k + C)$ |
| BFBSU | $n \cdot b$ | $n \cdot b \cdot (n \cdot k + C)$ | $b^n$ | $b^n \cdot (n \cdot k + C + R \cdot b^n)$ |

Notice that this time complexity considers the time it takes to create each node in addition to the number of nodes visited. This quantity (enclosed by parentheses) consists of the time to evaluate the utility function plus a constant $C$ for other data manipulating operations. For BFBSU, $R$ is a constant for the time it takes to do a single lookup in the conditional observation probability table with a worst case of $b^n$ elements in the table. We also note that the complexities for BFTE are multiplied by $k$ because of the $k$

instances of OPSAT; however, BFTE avoids expanding many of the search tree's fringe nodes by exploiting its *mutually preferential independent* [18] utility function, such that $b = 1$ in the best case.

This analysis shows that whenever $k > b$, the space required by BFBSE and BFBSU is always less than BFTE. Conversely, for large values of $n$, BFTE is faster in the best case by a factor of $b$ but both BFTE and BFBSE are equally fast in the worst case. In the worst case, BFBSU could potentially contain an exponential sized conditional probability table, but since most engineered systems do not have sensors that measure the entire system state, this term will remain close to linear in the number of components for real systems. In the following section, we will see that, for practical problems, $b$ is small and $C$ dominates over the utility function term unless the model is very large. For subsystem or modest size system models, BFBSE and BFBSU are more accurate, uses less memory, and requires less computation time than BFTE.

## 5.2 Experimental Results

The following empirical comparison between BFTE, BFBSE, and BFBSU was conducted using three different spacecraft models that are all roughly the size of a small subsystem. These models include Earth Observing One (EO-1) [11], Mars Entry Decent and Landing (EDL) system [13], and Space Technology 7 (ST7-A) [9]. All the algorithms were implemented in C++ and results were gathered using a 1.7GHz Intel Pentium M processor with 512MB of RAM.

### Earth Observing One

The EO-1 satellite was originally launched in November of 2000 as part of the New Millennium Program to validate instruments and technologies that will improve the performance of future Earth imaging observatories as well as reduce cost. Now in its extended mission, Livingstone 2 [15] was place on-board in September of 2004 and flight validated using models developed at NASA Ames. The model includes the Hyperion Imager, the Advanced Land Imager, the WARP data recording device, and other data transferring components. The model has a total of 60 variables, including 12 mode variables with an average domain size of 5.75.

#### Mars Entry Decent and Landing

The Mars EDL model was developed at MIT and contains all the major components used in the critical Mars entry, decent, and landing sequence. This includes a spacecraft propulsion system and a simplified navigation system. The model has a total of 42 variables, including 10 mode variables with an average domain size of 4.4.

#### Space Technology 7

The ST7-A model was developed as part of a NASA New Millennium concept definition study conducted by MIT and Johns Hopkins Applied Physics Laboratory. The ST7-A mission was to demonstrate autonomous science in low Earth orbit. The scaled down version of the ST7-A model used in this thesis includes a communication subsystem with a transmitter, switches, and redundant antennae. There are a total of 30 variables, including 8 mode variables with an average domain size of 3.5.

Table 5.2: Experimental Model Properties

| model | number of variables | number of mode variables | average mode variable domain size | number of clauses | average clause length |
|---|---|---|---|---|---|
| EO-1 | 60 | 12 | 5.75 | 195 | 2.39 |
| MarsEDL | 42 | 10 | 4.4 | 384 | 2.25 |
| ST7-A | 30 | 8 | 3.5 | 205 | 2.26 |

### 5.2.1 Accuracy

The single-step estimation scenario throughout Chapter 4 illustrated the likelihood-ordering limitations caused by the three major approximations made by BFTE, and the additional accuracy gained when eliminating two of the approximations in BFBSE and BFBSU. In addition, by considering estimation over many cycles (Figures 5-1, 5-2, and 5-3), the loss of belief state probability density for BFTE becomes readily apparent, highlighting the amount of state knowledge lost over time. The reduction in probability density is exponential in the number of estimation cycles for both BFBSE and BFBSU (as seen with the linear data for the semi-log plot on the right side of Figures 5-1, 5-2, and 5-3), but this reduction is dramatically less for BFBSE and BFBSU. Figure 5-1 shows the belief state probability density results for EO-1 when tracking 1, 10, and 30 estimates for 30 estimation cycles, under nominal observation and command sequences.

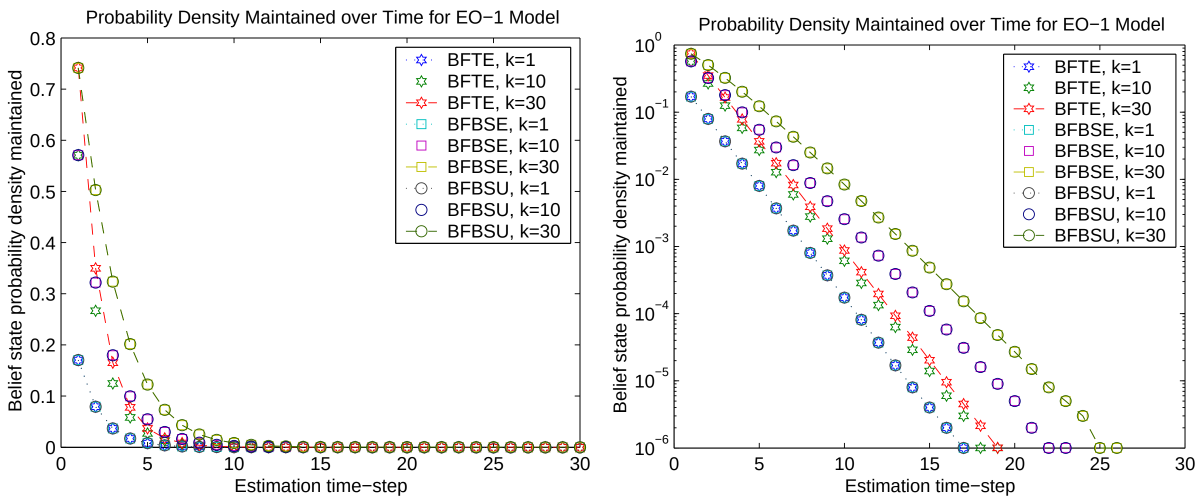


Figure 5-1: Probability density maintained over time for EO-1 model.

It is important to note that the loss in belief state probability density is identical for BFBSE and BFBSU, regardless of the number of estimates tracked over time. This is because the addition of observation probabilities does not change the amount of belief state probability consumed, it only adjusts the relative probabilities of the *a priori* estimates. All three estimation techniques maintain the same amount of belief state probability density when $k = 1$ since the utility function is the same for all of them when there is only one state in the approximate belief state.

Notice that the rate at which the belief state probability density is lost varies between models. The probability density quickly drops off for the EO-1 model and there is almost no loss at all when using BFBSE or BFBSU on the ST7-A model when $k \geq 10$. The rate at which the probability density is lost is directly related to how the probability is distributed across all the states in the belief state. As the number of reachable modes increases, or the relative difference between nominal and failure transition probabilities

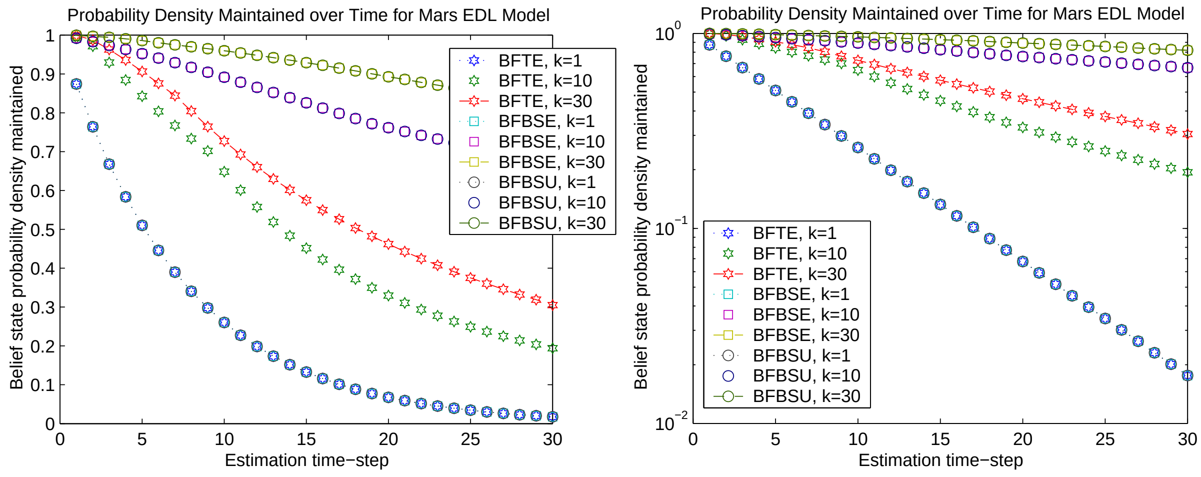


Figure 5-2: Probability density maintained over time for the Mars EDL model.

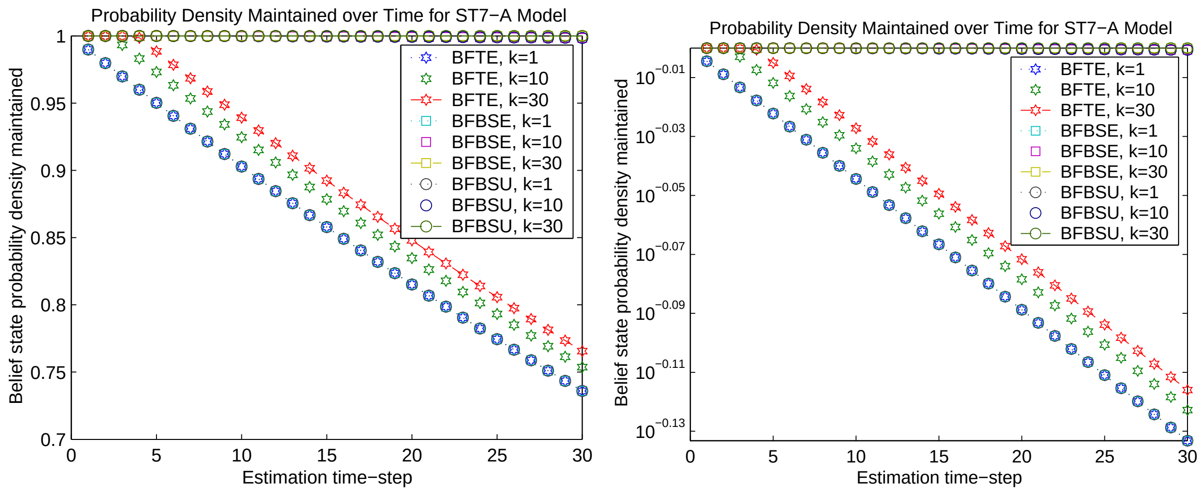


Figure 5-3: Probability density maintained over time for the ST7-A model.

decreases, the less concentrated the probability density will be. This is consistent with the large average domain size of mode variables for the EO-1 model, as previously shown in Table 5.2, since the probability is thinly distributed across many reachable modes. For more realistic models, the failure transition probabilities would also be much smaller, resulting in more probability concentrated in the leading estimates.

From the figures above, it is quite clear that BFBSE and BFBSU capture much more of the belief state probability density while tracking the same number of estimates as

BFTE. The less probability density lost, the more certain we can be about our estimates. Figure 5-4 focuses on two specific estimates within the belief state and compares the estimate probabilities between the three estimation techniques when $k = 10$. These two plots only use the ST7-A model since it is small enough to generate the exact solutions (indicated with circle marks) with only 512 enabled states. The plot on the left of Figure 5-4 shown the nominal state being tracked over time, and the right plot is a single-point failure state.

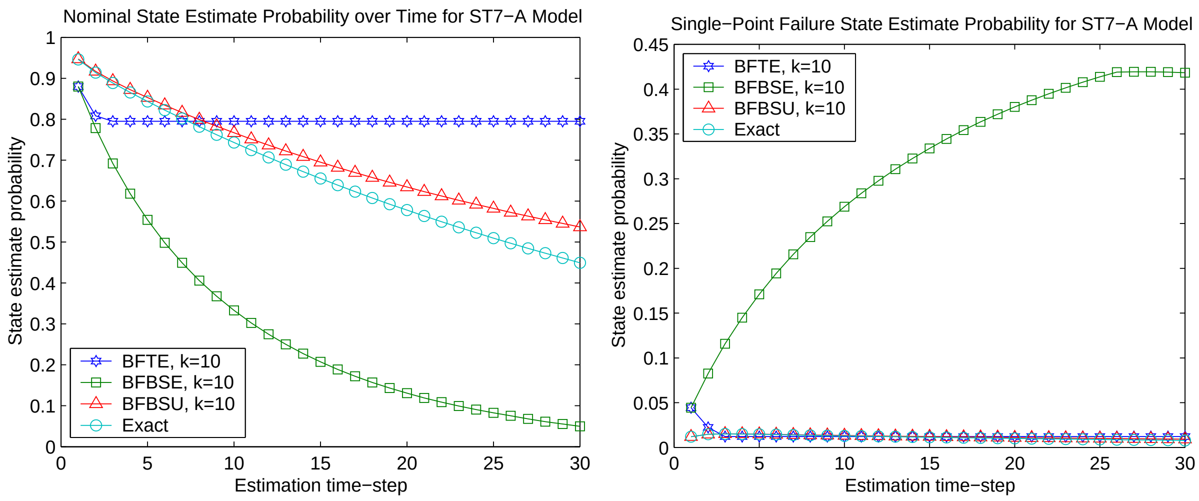


Figure 5-4: Single state estimate probability over time for the ST7-A model.

For both the nominal and failure states, the BFTE probability remains stagnant since it is continuously computing the same trajectory probabilities and then re-normalizing. This coincidentally follows the exact solution well for the failure mode but fails for the nominal mode. BFBSE tracks the model dynamics better but fails to udpate with the observation probabilities, leading to *a priori* trends. BFBSU tracks both nominal and failure modes closely since it uses the HMM belief state update equations directly as its utility function.

## 5.2.2 Performance

The space and time performance results are shown in Figures 5-5, 5-6, and 5-7 for EO-1, MarsEDL, and ST7-A, respectively. For a varying size initial belief state, space was measured by the maximum number of nodes placed in the A* priority queue, while estimation time was measured in milliseconds. Each estimation technique has two sets of data points: the solid lines represent the space and time required to generate the *single* best estimate from the $k$ states in the initial belief state (extracting best case behavior) and the dotted lines are the space and time required to generate the $k$ most likely estimates (simulating average case performance for the estimator in a real application).

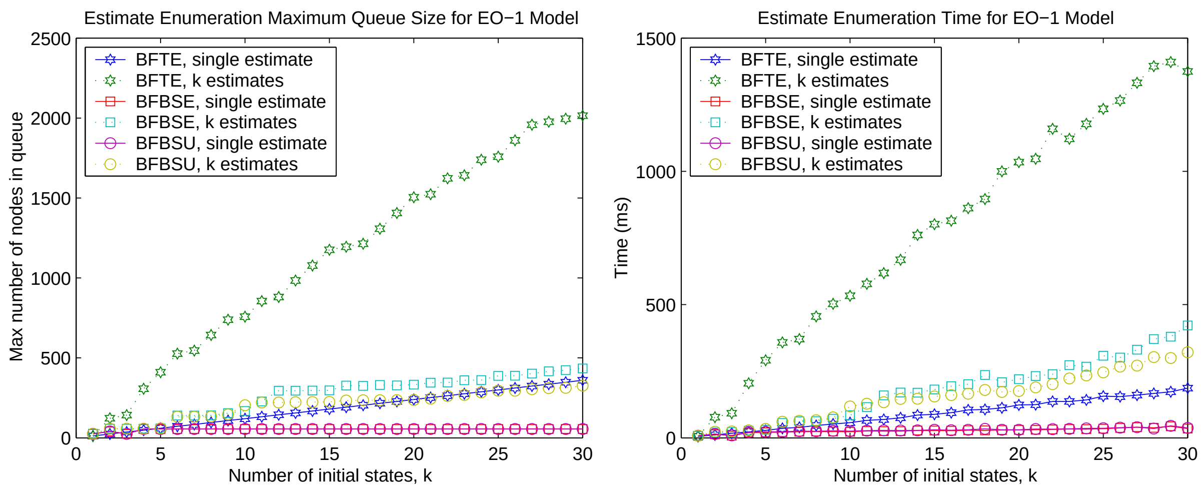


Figure 5-5: Best-case and average-case runtime performance for the EO-1 model.

The space and time performance results show good alignment with the complexity analysis. The single-estimate memory results for BFBSE and BFBSU (left side of the Figures 5-5, 5-6, and 5-7) reveal constant queue size, as predicted in the best case complexity analysis. Similarly, we see linear queue growth for BFTE. Comparing the $k$ estimate trends show that queue size is significantly less for BFBSE and BFBSU. Since Both BFBSE and BFBSU are framed as a single OCSP, their queue growth is identical in the best case and similar in the average case. For the EO-1 model in Figure 5-5, BFBSU has a smaller maximum queue size because its heuristic guided BFBSU to a different portion of the search space.

The time results (right side of Figures 5-5, 5-6, and 5-7) are also closely aligned to the complexity analysis. When enumerating only the single most likely state, BFTE is linear in $k$ and both BFBSE and BFBSU are nearly constant. This is because the

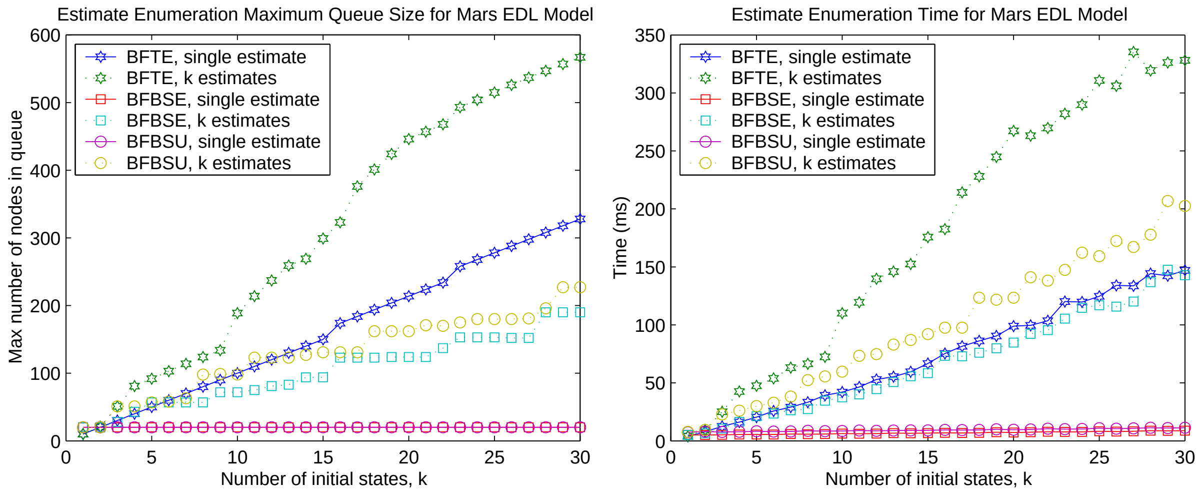


Figure 5-6: Best case and average case runtime performance for the Mars EDL model.

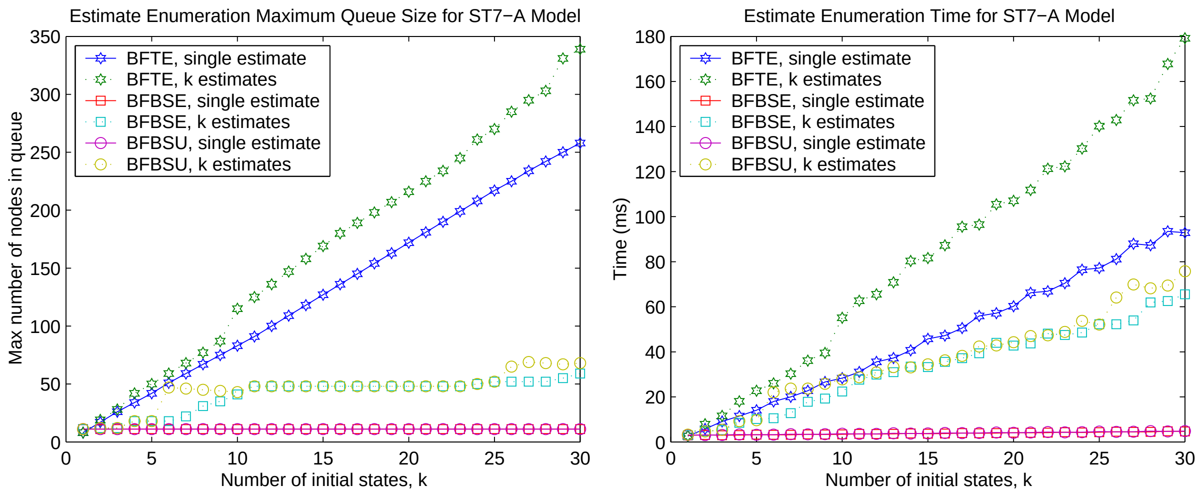


Figure 5-7: Best case and average case runtime performance for the ST7-A model.

arithmetic heuristic computation for each node is dominated by the constant term $C$ for these moderately sized models. The $k$ estimate trends show linear time increase in $k$, but BFBSE and BFBSU require significantly less computation time. BFBSU will always require at least as much time to execute as BFBSE when exploring the same nodes of the search tree. For the EO-1 model, BFBSU actually explored less nodes when searching for the $k$ most likely states (Figure 5-5), hence, took less time even though it had to trigger the observation probability rules for each node. Although the Mars EDL model has less components than EO-1, the observations in Mars EDL are tightly coupled to multiple components, resulting in more observation probability rules (shown in Table 5.3). The affects of this on runtime performance are shown in Figure 5-6.

Table 5.3: Number of Observation Probability Rules for each Model

| model | maximum # of OPRs | # OPRs required online |
|---|---|---|
| EO-1 | $1.77 \cdot 10^8$ | 64 |
| MarsEDL | $1.46 \cdot 10^6$ | 307 |
| ST7-A | $1.44 \cdot 10^4$ | 8 |

It is interesting to note that BFTE only outperforms BFBSE and BFBSU in both space and time when, as expected, $k < b$ where $b \approx 3$ for most real models. Since it is advantageous to set $k > 3$ (recall Figures 5-1, 5-2, and 5-3), BFBSE will outperform BFTE in space and time for moderate size models and sufficiently sized belief states (e.g., $k \approx 10$).

# Chapter 6

# Conclusion

This thesis presented *Best-First Belief State Enumeration (BFBSE)* and *Best-First Belief State Update (BFBSU)* as approximate monitoring and diagnosis techniques for Probabilistic Concurrent Constraint Automata (PCCA) that dramatically increases estimate accuracy, by *directly* using the Hidden Markov Model (HMM) belief state update equations. Since exact PCCA estimation consists of a belief state that is exponential in the number of components, previous approaches to mode estimation have made three significant approximations: (1) Belief state accuracy is maintained by only tracking the $k$ most likely states in the belief state. (2) State estimate probabilities are efficiently and accurately enumerated, by computing the $k$ most likely trajectories. (3) The PCCA observation probability distribution is simplified to 1 or 0, when the observations are consistent or inconsistent with a given state and model, without significant loss in estimation accuracy. BFBSE and BFBSU are novel in that they eliminate the last two of the three approximations to increase estimation accuracy, while requiring less memory and less computation time, when enumerating the approximate belief state for subsystem sized models.

BFBSE also enumerates estimates in best-first order, but removes the trajectory approximation by computing the estimate probabilities directly from the HMM propagation equation. This formulation allows for a more compact representation of the belief state and improves estimation accuracy. BFBSE is innovative in the way that the HMM propagation equation is used as a search guiding heuristic with a tight optimistic bound.

BFBSU extends BFBSE by updating the *a priori* state estimates with correct observation probabilities using both HMM belief state update equations. This is done efficiently by computing a compact set of observation probability rules that are quickly triggered online. BFBSU is novel in the way it applies the observation probabilities to the heuristic function to provide better diagnostic discrimination and avoid sub-optimal candidates.

## 6.1 Summary of Results

For BFBSE and BFBSU to be employed across a wide range of embedded systems, it is essential that they meet the stringent performance requirements imposed by these severely constrained computers. These challenges are exacerbated as future space exploration missions reach unprecedented levels of complexity. Our complexity analysis and empirical data shows that BFBSE and BFBSU outperform previous mode estimation techniques, requiring less memory and less computational time for subsystem sized models.

The most significant increase in computational performance comes from framing the PCCA estimation problem as a single OCSP. This reduces the size of the search space and removes redundant computations. BFBSE and BFBSU require additional heuristic computation over previous approaches, but this cost is insignificant due to savings gained from a more compact search space. In addition, BFBSU provides the search heuristic with a tighter optimistic bound that guides the search more efficiently toward the most likely estimates and avoids searching over sub-optimal candidates.

This thesis has introduced BFBSE and BFBSU as a mode estimation technique that significantly increases estimation accuracy, while reducing memory and computation requirements; providing an enabling monitoring and fault diagnosis technology for increasingly complex space missions of the future.

## 6.2 Future Work

Further improvements to PCCA mode estimation will result from addressing the following limitations: Since we use a factored HMM with concurrently operating components, we have made the assumption that all the components are independently transitioning. As highlighted in the IMU example of Section 3.1.2, the plant models are not always specified with independent transitions. In addition, the transition guards can occasionally be neither entailed or refuted, as mentioned in Section 3.1.1, which further violates the HMM propagation equation.

#### Mutually Inconsistent Modes

Recall the IMU plant scenario in Section 3.1.2. It is quite obvious that the IMU cannot be *Initializing* while the Power Switch is *Tripped Open* or *Open*, since the Power Switch does not supply the IMU with power in either of those modes. When framing PCCA estimation as an OCSP, the states that have these mutually inconsistent modes are identified as conflicts and are pruned out of the search space. Unfortunately, the transition probability leading into these conflicting states is $> 0$, and by simply removing the conflicting state, this probability goes unaccounted for. This results in the sum of the outgoing transitions to be incorrectly $< 1$.

One solution to this is to add up the transition probabilities for all the enabled transitions that lead to consistent target states, and normalize the outgoing transitions. This could be done efficiently by creating rules offline that identify when a state transition will lead to mutually inconsistent modes, and allow the estimator to correct (normalize) the transition probabilities prior to the best-first estimate search. Using the same approach described in Section 4.4, the dissents that have an empty antecedent, such that $\mathbf{o} = \emptyset$, identify which modes are mutually inconsistent. Recall that these dissents were not relevant for the observation probability computation because they were always conflicting, regardless of the observation assignments.

Although these transition probability rules would lead to a valid probability distribution, we still assume that re-normalizing the transition probabilities is valid. A more direct approach is to determine the mutually inconsistent modes during compile time, and combine the components that are directly coupled (such as the IMU and Power Switch). When combining the two components into one component, the new modes are the cross product of the mode assignments in each component and the mutually inconsistent modes are then excluded from the new component. This is very similar to the approach taken by Titan's Reactive Planner [3], and maintains the transition probabilities that were specified by the systems engineering who created the model.

### Consistent Transition Guards

As referred to in Section 3.1.1, it is possible for a transition guard to be consistent without being entailed or refuted. In order for the PCCA models to be probabilistically complete, there should be a probability associated with a transition guard that is merely consistent. The resulting component transition probability is shown in Equation 6.1.

$$\mathbf{P}(s_j^{t+1}|s_i^t, \mu^t) = \prod_{(x_a^{t+1}=v'_a)\in s_j^{t+1}} \big(\mathbf{P}(x_a^{t+1} = v'_a|x_a^t = v_a, g_a) \cdot \mathbf{P}(g_a)\big) \tag{6.1}$$

We can then take the same Maximum-Entropy approach that was used to define the observation probability distribution and assume a uniform probability distribution over the all possible consistent assignments to $g_a$. The resulting distribution is shown below:

$$\mathbf{P}(g_a) = \begin{cases} 1 & \text{if } s_i^t \wedge \mu^t \wedge \mathbb{Q} \models g_a, \\ 0 & \text{if } s_i^t \wedge \mu^t \wedge \mathbb{Q} \models \neg g_a, \\ 1/m & \text{otherwise,} \end{cases} \tag{6.2}$$

where $m$ = number of consistent assignments to $g_a$ for $s_i^t \wedge \mu^t \wedge \mathbb{Q}$.

# Appendix A

# IMU System Constraint Automata Definitions

The PCCA formalism was discussed in Section 2.2 and an example using the simple IMU system was provided; including a formal description of the IMU constraint automaton $\mathcal{A}_{imu}$ in Section 2.2.2. Appendix A provides formal descriptions for the remaining Power Switch $\mathcal{A}_{ps}$ and the Timer $\mathcal{A}_t$.

## A.1 The Power Switch Constraint Automaton, $\mathcal{A}_{ps}$

As described in Section 2.1.2, the Power Switch (PS) provides the IMU with power. For simplicity, we have assumed that the PS is always supplied with power and the power is transfered to the IMU when the switch is closed. Recall the graphical representation of $\mathcal{A}_{ps}$ shown again in Figure A-1. A formal description of $\mathcal{A}_{ps}$ is provided below:

1. $\Pi_{ps} = \{x_{ps}, \mu_{ps}^{cmd}, d_{ps}^{po}\}$ where $\{x_{ps}\} = \Pi_{ps}^{m}$ resides in 1 of 4 discrete modes $\mathbb{D}(x_{ps}) = \{$*op, cl, to, un*$\}$. $\Pi_{ps}^{r} = \{\mu_{ps}^{cmd}, d_{ps}^{po}\}$ where $\mu_{ps}^{cmd}$ is used to open and close the PS with $\mathbb{D}(\mu_{ps}^{cmd}) = \{$*open, close, no-command*$\}$ and $d_{ps}^{po}$ is the power-out with $\mathbb{D}(d_{ps}^{po}) = \{$*zero, nominal*$\}$ (same domain as $d_{imu}^{pi}$). $\Sigma_{ps} = \Sigma_{ps}^{m} \times \Sigma_{ps}^{r}$ is the set of all full assignments over $\Pi_{ps}$ with $4 \cdot (3 \cdot 2) = 24$ elements.

2. $\mathbb{M}_{ps}$ are the modal constraints for the PS, shown below in Table A.1.

3. The complete set of transition functions for $\mathcal{A}_{ps}$ are shown in Table A.2 below.

4. The component transition probability distribution for each set of PS transition functions is shown on the right side of Table A.2.

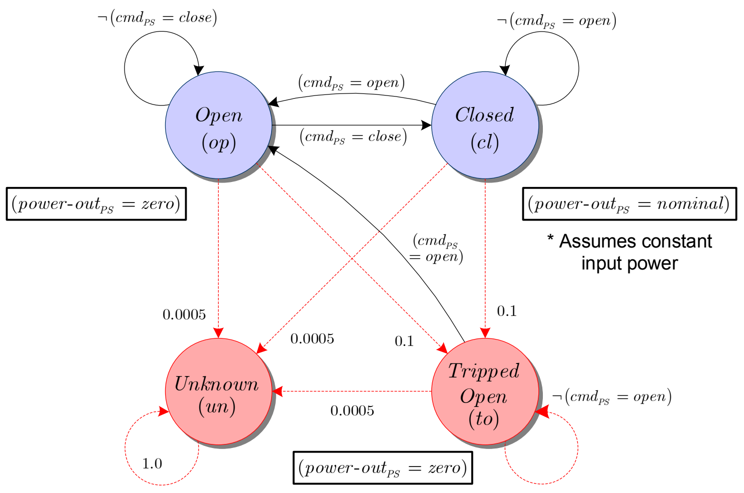


Figure A-1: PS constraint automaton, $\mathcal{A}_{ps}$.

Table A.1: $\mathcal{A}_{ps}$ Modal Constraints

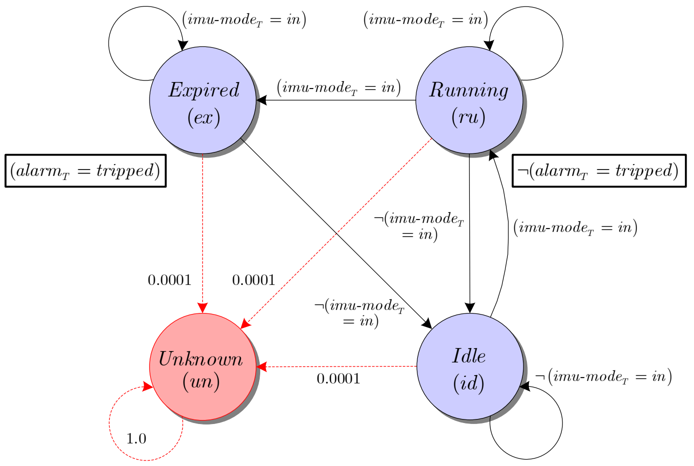
| $(x_{ps} = v_{ps}) \in \Sigma_{ps}^{x_{ps}}$ | $\mathbb{M}_{ps}(x_{ps} = v_{ps})$ |
|---|---|
| $x_{ps} = op$ | $d_{ps}^{po} = zero$ |
| $x_{ps} = cl$ | $d_{ps}^{po} = nominal$ |
| $x_{ps} = to$ | $d_{ps}^{po} = zero$ |
| $x_{ps} = un$ | (unconstrained) |

Table A.2: $\mathcal{A}_{ps}$ Transition Functions / Probabilities

| $(x_{ps} = v_{ps})$ | $g_{ps} \in \mathbb{C}(\Pi_{ps}^{r})$ | $\mathbb{T}_{ps}(x_{ps} = v_{ps}, g_{ps})$ | $\mathbf{P}_{\mathbb{T}_{ps}}(x_{ps} = v_{ps}, g_{ps})$ |
|---|---|---|---|
| $x_{ps} = op$ | $\neg(\mu_{ps}^{cmd} = close)$ | $\{op, to, un\}$ | $\{0.8995, 0.1, 0.0005\}$ |
| $x_{ps} = op$ | $\mu_{ps}^{cmd} = close$ | $\{cl, to, un\}$ | $\{0.8995, 0.1, 0.0005\}$ |
| $x_{ps} = cl$ | $\neg(\mu_{ps}^{cmd} = open)$ | $\{cl, to, un\}$ | $\{0.8995, 0.1, 0.0005\}$ |
| $x_{ps} = cl$ | $\mu_{ps}^{cmd} = open$ | $\{op, to, un\}$ | $\{0.8995, 0.1, 0.0005\}$ |
| $x_{ps} = to$ | $\neg(\mu_{ps}^{cmd} = open)$ | $\{to, un\}$ | $\{0.9995, 0.0005\}$ |
| $x_{ps} = to$ | $\mu_{ps}^{cmd} = open$ | $\{op, un\}$ | $\{0.9995, 0.0005\}$ |
| $x_{ps} = un$ | (unconstrained) | $\{un\}$ | $\{1\}$ |

## A.2 The Timer Constraint Automaton, $\mathcal{A}_t$

The Timer (T) was originally described in Section 2.1.3. The Timer starts when the IMU begins initializing and is used to determine if the IMU is stuck initializing. Recall the graphical representation of $\mathcal{A}_t$ shown again in Figure A-2.



$(imu\text{-}mode_T = in)$
$(imu\text{-}mode_T = in)$
*Expired* $(ex)$
$(imu\text{-}mode_T = in)$
*Running* $(ru)$
$(alarm_T = tripped)$
$\neg(alarm_T = tripped)$
$\neg(imu\text{-}mode_T = in)$
$(imu\text{-}mode_T = in)$
0.0001
0.0001
$\neg(imu\text{-}mode_T = in)$
*Unknown* $(un)$
*Idle* $(id)$
$\neg(imu\text{-}mode_T = in)$
0.0001
1.0

Figure A-2: T constraint automaton, $\mathcal{A}_t$.

A formal description of $\mathcal{A}_t$ is provided below:

1. $\Pi_t = \{x_t, o_t^{al}, d_t^{im}\}$ where $\{x_t\} = \Pi_t^m$ resides in 1 of 4 discrete modes $\mathbb{D}(x_t) = \{ex,\ ru,\ id,\ un\}$. $\Pi_t^r = \{o_t^{al}, d_t^{im}\}$ where $o_t^{al}$ is observation of the external continuous-time alarm monitor that can be *tripped* or *not-tripped* and $d_t^{im}$ is the current IMU mode with $\mathbb{D}(d_t^{im}) = \{of,\ in,\ me,\ si,\ un\}$ (as defined in Section 2.2.2). $\Sigma_t = \Sigma_t^m \times \Sigma_t^r$ is the set of all full assignments over $\Pi_t$ with $4 \cdot (2 \cdot 5) = 40$ elements.

2. $\mathbb{M}_t$ are the modal constraints for the T, shown below in Table A.3.

Table A.3: $\mathcal{A}_t$ Modal Constraints

| $(x_{ps} = v_t) \in \Sigma_t^{x_t}$ | $\mathbb{M}_t(x_t = v_t)$ |
|---|---|
| $x_t = ex$ | $o_t^{al} = tripped$ |
| $x_t = ru$ | $\neg(o_t^{al} = tripped)$ |
| $x_t = id$ | (unconstrained) |
| $x_t = un$ | (unconstrained) |

3. The complete set of transition functions for $\mathcal{A}_t$ are shown in Table A.4 below.

Table A.4: $\mathcal{A}_t$ Transition Functions / Probabilities

| $(x_t = v_t)$ | $g_t \in \mathbb{C}(\Pi_t^r)$ | $\mathbb{T}_t(x_t = v_t, g_t)$ | $\mathbf{P}_{\mathbb{T}_t}(x_t = v_t, g_t)$ |
|---|---|---|---|
| $x_t = ex$ | $d_t^{im} = in$ | $\{ex, un\}$ | $\{0.9999, 0.0001\}$ |
| $x_t = ex$ | $\neg(d_t^{im} = in)$ | $\{id, un\}$ | $\{0.9999, 0.0001\}$ |
| $x_t = ru$ | $d_t^{im} = in$ | $\{ex, ru, un\}$ | $\{0.49995, 0.49995, 0.0001\}$ |
| $x_t = ru$ | $\neg(d_t^{im} = in)$ | $\{id, un\}$ | $\{0.9999, 0.0001\}$ |
| $x_t = id$ | $d_t^{im} = in$ | $\{ru, un\}$ | $\{0.9999, 0.0001\}$ |
| $x_t = id$ | $\neg(d_t^{im} = in)$ | $\{id, un\}$ | $\{0.9999, 0.0001\}$ |
| $x_t = un$ | (unconstrained) | $\{un\}$ | $\{1\}$ |

4. The component transition probability distribution for each set of T transition functions is shown on the right side of Table A.4 above.

## A.3 Compiled IMU Plant Model Dissents

The dissent was defined in Definition 4.2 on Page 60 as a mapping from a partial assignment to mode variables to a conflict. This section lists all the dissents for the IMU Plant that are generated during offline model compilation:

$$(o_{imu}^{dv} = true) \Rightarrow \neg(x_{imu} = in) \qquad (o_t^{al} = tripped) \Rightarrow \neg(x_t = ru)$$

$$(o_{imu}^{dv} = true) \Rightarrow \neg(x_{imu} = si) \qquad (o_t^{al} = not\text{-}tripped) \Rightarrow \neg(x_t = ex)$$

$$(o_{imu}^{dv} = false) \Rightarrow \neg(x_{imu} = me)$$

$$(\ ) \Rightarrow \neg(x_{imu} = of \wedge x_{ps} = cl)$$

$$(\ ) \Rightarrow \neg(x_{imu} = in \wedge x_{ps} = op)$$

$$(\ ) \Rightarrow \neg(x_{imu} = in \wedge x_{ps} = to)$$

$$(\ ) \Rightarrow \neg(x_{imu} = si \wedge x_{ps} = op)$$

$$(\ ) \Rightarrow \neg(x_{imu} = si \wedge x_{ps} = to)$$

$$(\ ) \Rightarrow \neg(x_{imu} = me \wedge x_{ps} = op)$$

$$(\ ) \Rightarrow \neg(x_{imu} = me \wedge x_{ps} = to)$$

# Appendix B

# *Best-First Trajectory Enumeration*

Given the PCCA plant model $\mathcal{P}$, the current approximate belief state $\tilde{B}^t$, commands $\mu^t$, and resulting observations $o^{t+1}$, *Best-First Trajectory Enumeration (BFTE)* generates the estimates in the next belief state $\tilde{B}^{t+1}$ in best-first order according to the state trajectory probability $\mathbf{P}(s_j^{t+1} \mid s_i^t, \mu^t) \cdot p^t(s_i)$. BFTE leverages from all three approximations presented in Section 4.1 on Page 39 to achieve the performance requirements of full-scale embedded systems. This mode estimation technique was employed in Livingstone [21] and Livingstone 2 [15]. Livingstone was flight validated as part of the Remote Agent Experiment on-board Deep Space One in 1999 [17] and Livingstone 2 was flown on EO-1 in 2004. Pseudo code for the BFTE algorithm is shown in Figure B.1.

**Algorithm B.1** `BFTE`$(\mathcal{P}, \tilde{B}^t, \mu^t, o^{t+1})$

```
1: Setup the OCSP $\langle \mathbf{y}, f, C \rangle$:
     • The vector $\mathbf{x}$ includes a decision variable $\mathbf{x}_a$ for each component of the plant, whose domain $\mathbb{D}(\mathbf{x}_a)$ is the set of reachable target modes. The target mode for each transition $(x_a = v'_a) = \tau_a(x_a = v_a, g_a)$ is reachable when the source $(x_a = v_a) \in s_i^t$ and guard $g_a$ are satisfied by $C_M^t \wedge s_i^t \wedge \mu^t$. $C_M^t = \mathbb{Q} \wedge (\bigwedge_{(x_a=v_a)\in s_i^t} \mathbb{M}_a(x_a = v_a))$.
     • The utility function $f(\mathbf{x})$ is the prior trajectory probability of next state $\mathbf{x}$. More precisely, $f(\mathbf{x}) = \mathbf{P}(\mathbf{x} \mid s_i^t, \mu^t) \cdot p^t(s_i)$, where $\mathbf{P}(\mathbf{x} \mid s_i^t, \mu^t) = \prod_{(x_a=v'_a)\in\mathbf{x}} \mathbf{P}(x_a = v'_a \mid x_a = v_a, s_i^t, \mu^t)$ and $p^t(s_i)$ is the posterior probability for state $s_i^t$.
     • $C(\mathbf{y})$ encodes the constraint that $\mathbf{x} \wedge C_{M_\mathbf{x}} \wedge o^{t+1}$ must be consistent. $C_{M_\mathbf{x}} = \mathbb{Q} \wedge (\bigwedge_{(x_a=v'_a)\in\mathbf{x}} \mathbb{M}_a(x_a = v'_a))$.
2: Compute the $k$ most likely solutions $\tilde{S}^{t+1} = \{\mathbf{x^1}, \ldots, \mathbf{x^k}\}$ to $\wedge_{\mathbf{i}=1..k}\mathrm{OCSP}_\mathbf{i}\langle \mathbf{y}, f, C \rangle$ by comparing each next-best solution to the $k$ instances of OPSAT.
3: Extract the prior probabilities $p^{t+1} = \{f(\mathbf{x^1}), \ldots, f(\mathbf{x^k})\}$ for each solution $\mathbf{x^i} \in \tilde{S}^{t+1}$.
4: **return** the $k$ most likely trajectories contained by $\tilde{B}^{t+1} = \langle \tilde{S}^{t+1}, p^{t+1} \rangle$.
```

`BFTE` is a variation on the Viterbi algorithm [10], which calculates the most likely sequence of states for a HMM, given an initial state and a sequence of observations. The main difference between the algorithms is that BFTE only solves for the $k$ most likely sequences in best-first order, instead of computing the solution to all estimates in the belief state.

## B.1 BFTE Heuristic Function

In order for OPSAT to enumerate states in best-first order, an admissible heuristic must be specified for the OCSP utility function in `BFTE`. Recall the state trajectory probability equation shown again in Equation B.1. By separating the equation into a uniform-cost heuristic for the exact utility to partial assignments $n$, and a greedy heuristic as an optimistic utility for the remaining assignments, the BFTE heuristic function is provided in Equation B.2.

$$f(s_j^{t+1}) = \prod_{(x_a^{t+1}=v'_a)\in s_j^{t+1}} \big(\mathbf{P}(x_a^{t+1} = v'_a \mid x_a^t = v_a, s_i^t, \mu^t)\big) \cdot \mathbf{P}(s_i^t \mid o^{<0,t>}, \mu^{<0,t-1>}) \tag{B.1}$$

$$f(n) = \left( \begin{array}{ll} \prod_{(x_g^{t+1}=v'_g)\in n} & \big(\mathbf{P}(x_g^{t+1} = v'_g \mid x_g^t = v_g, s_i^t, \mu^t)\big) \cdot \\ \prod_{(x_h^{t+1}=v'_h)\notin n} & \max_{v'_h \in \mathbb{D}(x_h)} \big(\mathbf{P}(x_h^{t+1} = v'_h \mid x_h^t = v_h, s_i^t, \mu^t)\big) \cdot \\ & \mathbf{P}(s_i^t \mid o^{<0,t>}, \mu^{<0,t-1>}) \end{array} \right) \tag{B.2}$$

# Bibliography

[1] L. Baum and T. Petrie. Statistical inference for probabilistic functions of finite-state Markov chains. *Annals of Mathematical Statistics*, 37:1554–1563, 1966.

[2] R. E. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Transactions on Computers*, C-35(8):677–691, August 1986.

[3] S. Chung. A decomposed symbolic approach to reactive planning. Master's thesis, Massachusetts Institute of Technology, Department of Aeronautics and Astronautics, Cambridge, MA, May 2003.

[4] S. Chung, J. Van Eepoel, and B. C. Williams. Improving model-based estimation through offline compilation. In *Proceedings of ISAIRAS-01*, St-Hubert, Canada, June 2001.

[5] M. Cordier and C. Largouët. Using model-checking techniques for diagnosing discrete-event systems. In *Proceedings of DX-01*, pages 39–46, 2001.

[6] J. de Kleer and B. C. Williams. Diagnosing multiple faults. *Artificial Intelligence*, 32(1):97–130, 1987.

[7] J. de Kleer and B. C. Williams. Diagnosis with behavioral modes. In *Proceedings of IJCAI-89*, pages 1324–1330, Detroit, MI, 1989.

[8] P. Elliott. An efficient projected minimal conflict generator for projected prime implicate and implicant generation. Master's thesis, Massachusetts Institute of Technology, Department of Aeronautics and Astronautics, Cambridge, MA, February 2004.

[9] L. Fesq, M. Ingham, M. Pekala, J. Van Eepoel, D. Watson, and B. Williams. Model-based autonomy for the next generation of robotic spacecraft. In *Proceedings of 53rd Congress of the International Astronautical Federation*, Houston, TX, October 2002.

[10] G. D. Forney, Jr. The Viterbi algorithm. In *Proceedings of the IEEE*, pages 267–278, March 1978.

[11] S. C. Hayden, Adam J. Sweet, and Scott E. Christa. Livingstone model-based diagnosis of Earth Observing One. In *Proceedings of the AIAA Intelligent Systems*, 2004.

[12] R. Heninger. Mars Science Laboratory Proposal Information Package. Technical report, NASA Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA, April 14, 2004. D-27202 MSL-225-0502.

[13] M. Ingham. *Timed Model-based Programming: Executable Specifications for Robust Mission-Critical Sequences.* PhD thesis, Massachusetts Institute of Technology, Department of Aeronautics and Astronautics, Cambridge, MA, May 2003.

[14] E. T. Jaynes. On the rationale of maximum-entropy methods. In *Proceedings of the IEEE*, volume 70, pages 939–952, September 1982.

[15] J. Kurien and P. Nayak. Back to the future for consistency-based trajectory tracking. In *Proceedings of AAAI-00*, pages 370–377, 2000.

[16] O. Martin, M. Ingham, and B. Williams. Diagnosis as approximate belief state enumeration for probabilistic concurrent constraint automata. In *Proceedings of AAAI-05*, Pittsburgh, PA, July 9-13, 2005.

[17] P. Nayak, D. Bernard, G. Dorais, E. Gamble Jr., R. Kanefsky, J. Kurien, W. Millar, N. Muscettola, K. Rajan, N. Rouquette, B. Smith, W. Taylor, and Y. Tung. Validating the DS1 Remote Agent Experiment. In *Proceedings of ISAIRAS-99*, 1999.

[18] R. Ragno. Solving optimal satisfiability problems through Clause-directed A*. Master's thesis, Massachusetts Institute of Technology, Department of Aeronautics and Astronautics, Cambridge, MA, May 2002.

[19] P. Struss and O. Dressler. "Physical Negation" - Integrating fault models into the General Diagnostic Engine. In *Proceedings of IJCAI-89*, pages 1318–1323, Detroit, MI, 1989.

[20] B. C. Williams, M. D. Ingham, S. H. Chung, and P. H. Elliott. Model-based programming of intelligent embedded systems and robotic space explorers. *Proceedings of the IEEE*, 91(1):212–237, January 2003.

[21] B. C. Williams and P. Nayak. A model-based approach to reactive self-configuring systems. In *Proceedings of AAAI-96*, pages 971–978, Portland, OR, August 4-8, 1996.

[22] B. C. Williams and R. Ragno. Conflict-directed A* and its role in model-based embedded systems. *To appear in the Journal of Discrete Applied Math (accepted 2001)*, 2005.