# A TRACTABLE APPROACH TO PROBABILISTICALLY ACCURATE MODE ESTIMATION

**Oliver B. Martin, Seung H. Chung, and Brian C. Williams**

Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology
32 Vassar Street, Cambridge, MA 02139, U.S.A
{omartin, chung, williams}@mit.edu

## ABSTRACT

As space exploration missions grow increasingly complex and are required to operate over extended durations of time, there is an amplified demand for accurate autonomous health management systems. In response, model-based programming approaches have been developed as scalable solutions to autonomous health management. Previous model-based monitoring and diagnosis techniques, such as Livingstone, were demonstrated to successfully and efficiently track nominal and failure modes in an abundance of scenarios, but at the cost of making simplifying assumptions about the observation probability that can lead to erroneous diagnoses after extended operation. Extending on Best-First-Belief-State-Enumeration (BFSE), this paper presents a new mode estimation technique called *Best-First Belief State Update (BFBSU)* that eliminates the observation probability assumption. BFBSU uses the full two-stage HMM belief state update equations as its utility function, thus further increasing estimator accuracy, while maintaining the efficiency required for real-time monitoring and fault detection.

## 1. INTRODUCTION

Model-based inference reasons over a composition of interconnected subsystem components, modeled as a factored variant of a Hidden Markov Model (HMM) called Probabilistic Concurrent Constraint Automata (PCCA) [1]. For PCCA, a belief state is computed using the standard HMM belief state update equations [2] that require *a priori* knowledge of conditional observation probabilities. The observation probability distribution for PCCA was defined in [3, 1] but is difficult to calculate due to the large state space of sensory observations and component modes. As a result, many previous model-based diagnosis techniques assume the observation probability to be 1 or 0 depending on if the observation is simply consistent or inconsistent with the projected state. For a failure mode with one or more consistent observations, the total observation probability space is incorrectly $\geq 1$ and results in a probabilistic bias toward the failure mode, eventually leading to an incorrect fault diagnosis.

Our approach eliminates the observation probability assumptions, 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 probability rules (OPRs) that map system state to observation probabilities. Then, during each online estimation cycle, the appropriate OPRs are triggered and used to compute the HMM belief state update equations.

We incorporate the correct observation probabilities within Best-First Belief State Estimation (BFBSE) [4] 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 show that, under certain conditions, BFBSU will outperform BFBSE in time and memory.

### 1.1. PCCA Overview

*Probabilistic Concurrent Constraint Automata* (PCCA) [1] 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. Formally, a probabilistic constraint automaton for component "$a$" is defined by the tuple $\mathcal{A}_a = \langle \Pi_a, \mathbb{M}_a, \mathbb{T}_a, \mathbf{P}_{\mathbb{T}_a} \rangle$:

*Proc. of 'The 8th International Symposium on Artifical Intelligence, Robotics and Automation in Space - iSAIRAS', Munich, Germany. 5-8 September 2005, (ESA SP-603, August 2005)*

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 $\lambda \triangleq \mathbf{True} \mid \mathbf{False} \mid (u = y) \mid \neg\lambda_1 \mid \lambda_1 \wedge \lambda_2 \mid \lambda_1 \vee \lambda_2$, 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 $\Pi^d \subseteq \bigcup_{a=1..n} \Pi_a^r$. $\Sigma^m$, $\Sigma^c$, $\Sigma^o$, and $\Sigma^d$ are the sets of full assignments over $\Pi^m$, $\Pi^c$, $\Pi^o$, and $\Pi^d$.
3. $\mathbb{Q} \subset \mathbb{C}(\Pi)$ is a set of finite domain constraints that capture the interconnections between plant components.

### 1.2. Belief State Update Overview

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 $p(s_i) \in p$.

The Markov 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 (Eqs. 1 and 2) [2].

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

$$\mathbf{P}(s_j^{t+1} | o^{<0,t+1>}, \mu^{<0,t>}) = \frac{\mathbf{P}(s_j^{t+1} | o^{<0,t>}, \mu^{<0,t>}) \cdot \mathbf{P}(o^{t+1} | s_j^{t+1})}{\sum_{s_i^{t+1} \in S^{t+1}} \mathbf{P}(s_i^{t+1} | o^{<0,t>}, \mu^{<0,t>}) \mathbf{P}(o^{t+1} | s_i^{t+1})} \quad (2)$$

Eq. 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. Eq. 1 propagates the system dynamics into the future before considering new observations. Once all the *a priori* estimates are generated, Eq. 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 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 [3], such that for every state $s_j \in \Sigma^m$, there is a probability distribution across all combinations of observations (Eq. 3). 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 [5] when there is no previous knowledge about how the sensors behave.

$$\mathbf{P}(o^{t+1} | s_j^{t+1}) = \begin{cases} 1 & \text{if } s_j^{t+1} \wedge \mathbb{M} \models o^{t+1}, \\ 0 & \text{if } s_j^{t+1} \wedge \mathbb{M} \models \neg o^{t+1}, \\ 1/m & \text{otherwise,} \end{cases} \quad (3)$$

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

This observation probability, however, 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 [6, 7] simplified the observation probability distribution further by assuming the observation probability as 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.

### 1.3. PCCA Estimation as an Optimal Constraint Satisfaction Problem

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 [1] and can similarly be used to formulate the methods underlying GDE [3], Sherlock [8], and Livingstone [6, 7]. We leverage a similar Optimal Constraint Satisfaction Problem (OCSP) [9] formulation.

**Definition 1** *An OCSP $\langle \mathbf{y}, f, C \rangle$ is a problem of the form "arg max $f(\boldsymbol{x})$ subject to $C(\mathbf{y})$," where $\boldsymbol{x} \subseteq \mathbf{y}$ is a vector of decision variables, $C(\mathbf{y})$ is a set of state constraints, and $f(\boldsymbol{x})$ is a multi-attribute utility function.*

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 1.1 provides pseudo code for computing the exact belief state update when framing PCCA estimation as an OCSP.

Algorithm 1.1 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 (Eq. 2). This procedure is repeated for each estimation cycle.

**Algorithm 1.1** 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 $B^{t+1} = \langle S^{t+1}, p^{t+1} \rangle$.

## 2. 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 Eq. 3. Compactness is essential for BFBSU in order to be scalable to full-sized systems.

**Definition 2** *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 $\bar{\mathbf{o}}$. There are two assumptions that make this claim legitimate: Given the PCCA observation probability distribution defined in Eq. 3, 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 [1], an instance of OCSP solver, 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.

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 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 $|\mathbb{D}(x_a)|$.

## 3. 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 [11, 12]. 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 3** *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 3.1.

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.

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 to compute the observation probability for the subset of observation variables that were connected in the subgraph. Since the OPR antecedents in each

[1] An OCSP solver based on *Conflict-directed A** [9], which efficiently finds solutions to an OCSP in best-first order, by interleaving candidate generation and test.

**Algorithm 3.1** 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

CPT are mutually exclusive, only one OPR from each CPT can be triggered at once. 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 $\prod_{o_i \in \mathbf{o}} |\mathbb{D}(o_i)|$.

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.

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.

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 [3]. 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) [13]. 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.

**Algorithm 4.1** 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 $\tilde{B}^{t+1} = \langle \tilde{S}^{t+1}, p^{t+1} \rangle$.

## 4. 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 [12]. Accuracy of online mode estimation is increased by extending BFBSE [4] 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 4.1.

It is important to note that the BFBSU OCSP formulation is nearly identical to the exact PCCA estimation formulation in Alg. 1.1, except that we only track the $k$ most likely estimates in an approximate belief state $\tilde{B}$. BFBSU is an improvement over BFBSE because, in addition to using the HMM propagation equation (Eq. 1), BFBSU

also folds in the HMM update equation (Eq. 2) directly into its utility function.

A tractable approach to generating a compact set of observation probability rules provide fast online heuristic computation. The heuristic function for the BFBSU *Conflict-directed A** search is provided in Eq. 4. Eq. 5 is the combined form of the HMM belief state update equations without the normalization factor in the update step. The BFBSU heuristic function (Eq. 4) 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.

$$f(n) = \Sigma_{s_i^t \in \tilde{S}^t} \left( \begin{array}{l} \Pi_{(x_g^{t+1}=v'_g)\in n} \\ \quad \left(\mathbf{P}(x_g^{t+1} = v'_g \mid x_g^t = v_g, s_i^t, \mu^t)\right) \\ \Pi_{(x_h^{t+1}=v'_h)\notin n} \max_{v'_h \in \mathbb{D}(x_h)} \\ \quad \left(\mathbf{P}(x_h^{t+1} = v'_h \mid x_h^t = v_h, s_i^t, \mu^t)\right) \\ \mathbf{P}(s_i^t \mid o^{<0,t>}, \mu^{<0,t-1>}) \end{array} \right) \cdot \mathbf{P}(o^{t+1} \mid n) \quad (4)$$

$$\mathbf{P}(s_j^{t+1} | o^{<0,t+1>}, \mu^{<0,t>}) = \Sigma_{s_i^t \in \tilde{S}^t} \left( \begin{array}{l} \Pi_{(x_a^{t+1}=v'_a)\in s_j^{t+1}} \\ \quad \left(\mathbf{P}(x_a^{t+1} = v'_a \mid x_a^t = v_a, s_i^t, \mu^t)\right) \\ \mathbf{P}(s_i^t \mid o^{<0,t>}, \mu^{<0,t-1>}) \end{array} \right) \cdot \mathbf{P}(o^{t+1} \mid s_j^{t+1}) \quad (5)$$

A tight bound on the heuristic function is important because it prevents the search from enumerating highly suboptimal 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.

## 5. RESULTS

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 $|\mathbb{D}(x_a)|$, $n$ is the number of components in the model $|\Pi^m|$, and $k$ is the number of belief states being tracked. BFBSE and BFBSU have the same best and worst space complexity, $n \cdot b$ and $b^n$, respectively, and the same best case time complexity[2], $n \cdot b \cdot (n \cdot k + C)$.

For the worst case, however, the time complexity for BFBSE is $b^n \cdot (n \cdot k + C)$ and for BFBSU is $b^n \cdot (n \cdot k + C + R \cdot b^n)$. 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.[3] 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.

### 5.1. Experimental Results

The following empirical comparison between 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) [14], Mars Entry Decent and Landing (EDL) system [10], and Space Technology 7 (ST7-A) [15]. 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 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 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 has a a total of 30 variables, including 8 mode variables with an average domain size of 3.5.

[2]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.

[3]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.

*Table 1. 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 |

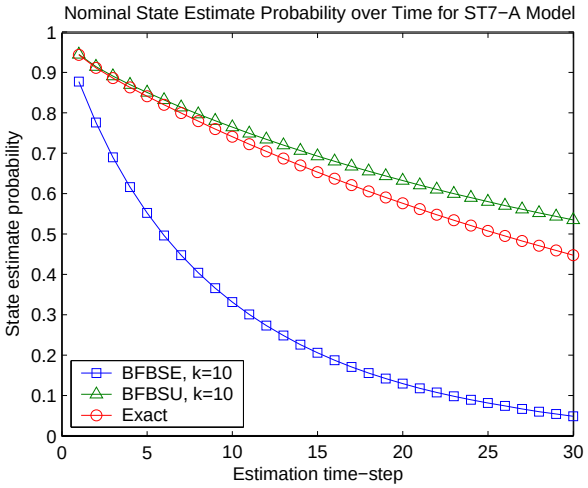


*Figure 1. Single nominal state estimate probability over time for the ST7-A model.*

Table 1 lists the number of OPRs for each of these models. Note that the actual number of OPRs required is multiple orders of magnitude less than the worst case.

### 5.2. Accuracy

Figs. 1 and 2 respectively focuses on a specific nominal state and a failure state estimates within the belief state over time and compares the estimate probabilities between the two 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. BFBSE tracks the model dynamics but fails to update 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.

### 5.3. Performance

The space and time performance results are shown in Figs. 3 and 4, respective, for ST7-A. 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).

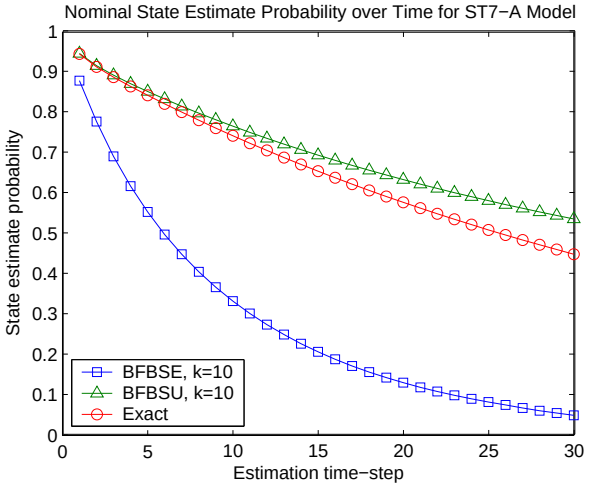


*Figure 2. Single failure state estimate probability over time for the ST7-A model.*

The space and time performance results show good alignment with the complexity analysis. The single-estimate memory results for BFBSE and BFBSU (Fig. 3) reveal constant queue size, as predicted in the best case complexity analysis. Comparing the $k$ estimate trends show that the growth of the queue size is identical in the best case and similar in the average case. For certain models, such as the EO-1, BFBSU has a smaller maximum queue size because its heuristic guided BFBSU to a different portion of the search space. The time results (Fig. 4) are also closely aligned to the complexity analysis. When enumerating only the single most likely state, both BFBSE and BFBSU are nearly constant. This is because the 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 $k$ for both BFBSE and BFBSU.

## ACKNOWLEDGMENTS

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.

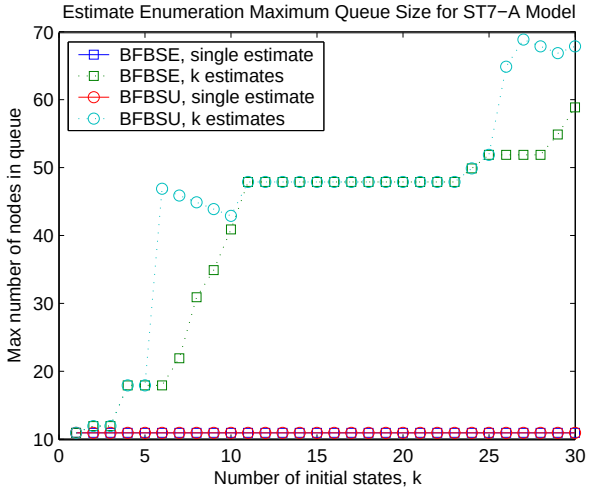


*Figure 3. Best case and average case runtime queue size for performance for the ST7-A model.*

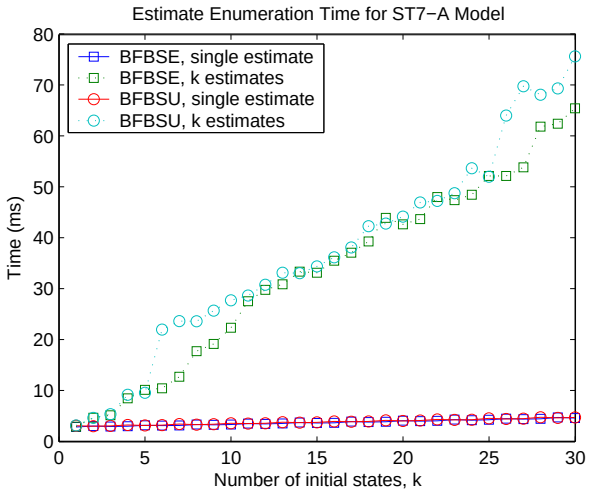


*Figure 4. Best case and average case time performance for the ST7-A model.*

## REFERENCES


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

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

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

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

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

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

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

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

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

10. Ingham, M. *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.

11. Elliott, P. 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.

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

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

14. Hayden, S. C., Sweet, A. J., and Christa, S. E. Livingstone model-based diagnosis of Earth Observing One. In *Proceedings of the AIAA Intelligent Systems*, 2004.

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