Computer Science and Artificial Intelligence Laboratory
Technical Report

MIT-CSAIL-TR-2018-016 May 24, 2018

Best-first Enumeration Based on Bounding Conflicts, and its Application to Large-scale Hybrid Estimation

Eric Timmons and Brian C. Williams

massachusetts institute of technology, cambridge, ma 02139 usa — www.csail.mit.edu

Best-first Enumeration Based on Bounding Conflicts, and its Application to Large-scale Hybrid Estimation

Eric Timmons ETIMMONS@MIT.EDU
Brian C. Williams WILLIAMS@MIT.EDU
MIT Computer Science and Artificial Intelligence Laboratory
32 Vassar St
Cambridge, MA 02139 USA

Abstract

With the rise of autonomous systems, there is a need for them to have high levels of robustness and safety. This robustness can be achieved through systems that are self-repairing. Underlying this is the ability to diagnose subtle failures. Likewise, online planners can generate novel responses to exceptional situations. These planners require an accurate estimate of state. Estimation methods based on hybrid discrete/continuous state models have emerged as a method of computing precise state estimates, which can be employed for either diagnosis or planning in hybrid domains. However, existing methods have difficulty scaling to systems with more than a handful of components. Discrete state estimation capabilities can scale to this level by combining best-first enumeration and conflict-directed search. Best-first methods have been developed for hybrid estimation, but the creation of conflict-directed methods has previously been elusive. While conflicts are used to learn from constraint violation, probabilistic hybrid estimation is relatively unconstrained. In this paper we present an approach to hybrid estimation that unifies best-first enumeration and conflict-directed search through the concept of "bounding" conflicts, an extension of conflicts that represent tighter bounds on the cost of regions of the search space. This paper presents a general best-first search and enumeration algorithm based on bounding conflicts (A*BC) and a hybrid estimation method based on this enumeration algorithm. Experiments show that an A*BC powered state estimator produces estimates faster than the current state of the art, particularly on large systems.

1. Introduction

There is a continuously growing demand for complex systems with autonomous decision making capabilities that are robust and safe. This robustness can be achieved using systems that have the ability to self-repair and use online planners to generate novel responses to exceptional situations. A key capability needed by such systems is the ability to accurately estimate the state of the system. While discrete models have long been a mainstay of the model-based reasoning community (De Kleer & Williams, 1987), these models do not have the requisite resolution needed when controlling or detecting incipient failures in highly dynamic systems.

For example, consider electrical and fluid systems on naval ships (Srivastava, Cartes, Maturana, Ferrese, Pekala, Zink, Meeker, Carnahan, Staron, Scheidt, & Huang, 2008). Such systems consist of components such as valves, pipes, pumps, and electrical relays. Any of these components can become damaged: pipes can be destroyed, valves can stick closed, and so on. Additionally, components can have complex continuous state behaviors, such as spin up/down time for pumps or heat transfer between electrical loads, cold water heat exchangers, and the environment. Any planning and execution framework or operator using these systems would like to quickly determine both the continuous and discrete state in order to determine if a component is damaged, confirm that commanded transitions actually happen, and determine a command sequence to reach a target state. However, the sensors in the system typically measure only continuous quantities such as temperature, flow rate, and pressure; the discrete state of the system has to be inferred from the continuous measurements and knowledge of the, sometimes subtle, differences in continuous behavior induced by different operating modes. Further examples in the same vein include control of unmanned spacecraft (Muscettola, Nayak, Pell, & Williams, 1998) and power supply restoration (Thiébaux, Cordier, Jehl, & Krivine, 1996).

While exact hybrid estimation is theoretically simple — given an appropriate continuous state estimator, generate a continuous state estimate for every possible discrete state trajectory — it quickly becomes infeasible because the number of discrete trajectories grows exponentially with time. As such, the current state of the art in hybrid state estimation focuses on approximate estimation. These techniques include Multiple Model (MM) methods such as the Generalized Pseudo-Bayesian Algorithm (GPB) (Ackerson & Fu, 1970), the detection-estimation method (Tugnait, 1982), the residual correlation Kalman filter bank (Hanlon & Maybeck, 2000), the Interacting Multiple Model (IMM) algorithm (Blom & Bar-Shalom, 1988), and adaptive MM methods by Li et al. (1996, 1999, 2000). More recently, techniques such as the Hybrid Mode Estimator (HME) (Hofbaur & Williams, 2002, 2004), the Hybrid Diagnostic Engine (HyDE) (Narasimhan & Brownston, 2007), and combined stochastic/greedy estimation (Blackmore, Funiak, & Williams, 2008) have also been developed.

While these state of the art techniques have been shown to effectively estimate the hybrid state of small subsystems comprised of a hand-full of components, they have difficulties scaling to larger, real-world systems. In this paper, we introduce an extension of HME which uses a novel search algorithm (A*BC) that reduces the number of discrete states that need to be explored in order to discover the most likely states of the system. Essential to this approach is the generalization of conflict-directed best-first search to hybrid domains, through the concept of bounding conflicts.

Purely discrete state estimators from the model-based reasoning community are able to scale in large part through the use of best-first enumeration and conflict-directed search (Williams & Ragno, 2007), as well as stochastic search methods (Feldman, Provan, & Van Gemund, 2010). Conflict-directed search serves the role of efficiently pruning large sets of inconsistent states (i.e., states with zero probability), best-first enumeration focuses the estimator on the states that are most likely, and stochastic methods allow the algorithms to remove the burden of completeness. While all methods are important, conflicts have been shown to be particularly effective due to their pruning ability. While scaling of hybrid estimation methods has been improved through best-first (Hofbaur & Williams, 2002) and sampling-based methods (Blackmore et al., 2008), the creation of effective conflict-directed methods has proven more challenging. The technical challenge is that a conflict is a consistency based concept and represents sets of states that have zero probability based on a proof of logical inconsistency. However, in a stochastic environment with unbounded uncertainty (such as Gaussian noise models), all behaviors are consistent, albeit unlikely.

It is possible to incorporate consistency-based conflicts into hybrid estimation by placing thresholds on estimate probabilities. If the probability of an estimate falls below this threshold, the corresponding mode is considered to be infeasible and a conflict is learned. While this approach can produce more precise estimates than an approach using a discretization of the system, it requires the operator to explicitly trade off diagnostic accuracy for scalability. In order to decrease the time needed to find any estimates, the operator would want to push the threshold as close as possible to the nominal value. This increases the number of estimates falling beyond the threshold, resulting in more conflicts learned and more of the search space being ignored. However, this reduces the precision of the estimation and makes it harder to respond to small, unlikely disturbances that fall just outside the threshold. Conversely, in order to increase precision, the operator would want to move the threshold as close to zero as possible. The problem with this is that extremely few conflicts will be learned, so there is little to no improvement over not using conflicts.

An ideal hybrid state estimation algorithm would be an any-time algorithm that quickly enumerates the most likely estimates of the system and, given enough time, would produce all feasible estimates. In this paper we present such an approach to hybrid estimation that augments best-first enumeration with a variant of conflicts that is more suitable to probabilistic, rather than consistency-based, inference through the concept of *bounding conflicts*. A bounding conflict replaces the concept of a compact encoding of inconsistent candidates with the concept of a compact encoding of an area of the search space where the predicted cost — such as one obtained from a heuristic bounding function — is much lower than the true cost. Bounding conflicts are similar in spirit to *valued no-goods* (Dago & Verfaillie, 1996), but more powerful as bounding conflicts provide a tighter bound function instead of a static bound.

While the focus of this paper is on hybrid estimation, we gain insight into bounding conflicts by viewing hybrid estimation as an instance of an optimization problem. Specifically, for constrained optimization algorithms, such as conflict-directed A*, traditional conflicts are learned from the set of constraints of the problem while the search is ongoing. Bounding conflicts, in contrast, are learned from the problem's objective function, and a bounding function of that objective. This offers insight as to how to effectively generalize conflict-directed search from satisfaction problems to optimization problems. This paper presents both a general best-first enumeration algorithm based on bounding conflicts, and a hybrid estimation method based on this general capability. For hybrid estimation, this allows the search for mode assignments to learn which mode assignments are unlikely, given the observations and avoid them to quickly focus in on the best candidates. However, these poor mode assignments aren't discarded, instead they are saved for later, to be expanded and tested if there is enough time.

We first review the definitions for concurrent probabilistic hybrid automata (CPHAs) — the hybrid model used for the remainder of the paper — and provide an example hybrid system that will be used as a running example. Second, we define the hybrid state estimation problem for CPHAs and develop a state estimator based on best-first enumeration. Third, we develop the A* with Bounding Conflicts (A*BC) search algorithm, the key innovative component of the best-first state estimator. Last, we provide empirical data showing that A*BC enables the hybrid state estimator to provide estimates for real world problems that cover significantly more probability mass in less time as compared to a state of the art, non-A*BC powered algorithm.

2. Hybrid System Modeling

Hybrid state estimation is a process that takes as input a probabilistic model of the hybrid discrete/continuous system under observation together with a sequence of observations and commands up to time t, and computes the belief state of the system at time t. In this section we review the hybrid modeling scheme that this paper employs and then define hybrid estimation for this model in the subsequent section.

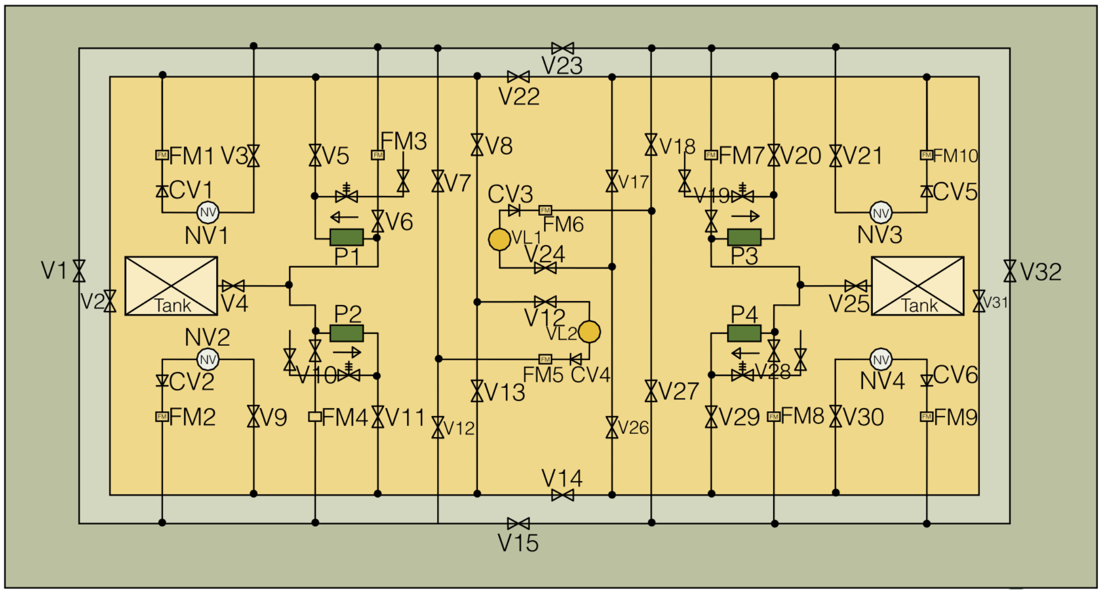


Figure 1: Example hybrid system. The pumps are labeled P1-P4 and the loads are labeled NV1-NV4 (Non-Vital load) and VL1-VL2 (Vital Load). The valves are labeled V1-V32 and check-valves are labeled CV1-CV6. The flow meters are labeled FM1-FM10.

In this work, we focus on systems with a large number of discrete variables, srepresenting a set of concurrent processes. As such, we use concurrent probabilistic hybrid automata (CPHA) (Hofbaur & Williams, 2004) to model systems. CPHAs are a compact way of modeling real world systems composed of interacting components, each with a set of discrete and continuous state variables. CPHAs are particularly useful at modeling systems where each component's discrete state describes its *operating mode* and the component's continuous dynamics (and noise characteristics) are determined by the component's mode.

As a motivating example, consider a naval ship's electrical and cooling system, described by Srivastava, et al. (2008). The system, pictured in Figure 1, consists of a series of electrical loads and a fluid system to cool the loads. There are four pumps and six heat exchangers for the loads, with valves capable of routing water from pumps to the active exchangers.

Using CPHAs, this system can be compactly represented using eighty-eight components (twenty-eight T-junctions, two tanks, four pumps, four non-vital loads, two vital loads, six check valves, thirty-two valves, and ten flow meters). Twenty-eight of the components (the T-junctions) have only one mode, the remaining components have an average of three modes (for example, the valves can be open, closed, stuck open, and stuck closed), meaning the system has over 10^{28} possible discrete configurations it can be in.

Under normal operating conditions, the pumps are separated from each other by closed valves and each load is serviced with water by a single pump. However, the system may become damaged during operation, possibly necessitating that non-vital loads be shed and valves be reconfigured to restore water supply to vital loads. These decisions could be made by a control program that relies on a state estimator to determine the likely configuration of the system.

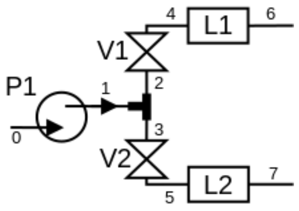


Figure 2: Running example.

As a pedagogical example, we present a simplified version of this cooling system, pictured in Figure 2. The example system consists of a pump, two valves, two loads with heat exchangers, and one T-junction (not labeled). The pump is connected to a tank of chilled water and the loads empty into a tank. The pump is an idealized pressure source that, when on, provides any amount of flow necessary to maintain a constant pressure. When off, there is no flow through the pump. The valves and loads provide some resistance to flow through them. The following subsection introduces detailed hybrid models for the loads.

2.1 Probabilistic Hybrid Automata

Concurrent probabilistic hybrid automata model large-scale hybrid systems at two levels: component and system. The component level model is a *probabilistic hybrid automaton (PHA)*, and is used to describe each component of the system in isolation. A PHA specifies a component's operating modes, its state variables, its interfaces to other automata and the world, the evolution of its continuous state in each operating mode, and the evolution of its operating mode.

Definition 1. *A probabilistic hybrid automaton (PHA) is described by the 7-tuple* $\langle \mathbf{x}, \mathbf{w}, F, T, \mathcal{X}_d, U_d, T_s \rangle$*, where:*

- $\mathbf{x} = x_d \cup \mathbf{x}_c$ is the set of the automaton's state variables. This set consists of a discrete state variable $x_d \in \mathcal{X}_d$ that describes the automaton's operating mode, as well as a set of continuous state variables, $\mathbf{x}_c \in \mathbb{R}^{n_x}$.
- The set $\mathbf{w} = \{\mathbf{u}_d, \mathbf{w}_c, \mathbf{v}_c\}$ denotes the interface variables of the automaton. These variables are the mechanism by which the environment and other automata interact with this automaton. It consists of discrete valued command (input) variables $\mathbf{u}_d \in U_d$, continuous interface variables (that can serve as input or output) $\mathbf{w}_c \in \mathbb{R}^{n_w}$, and Gaussian noise input variables $\mathbf{v}_c \in \mathbb{R}^{n_v}$.
- $F : \mathcal{X}_d \rightarrow 2^{\mathcal{F}_{DE} \cup \mathcal{F}_{AE}}$ specifies the continuous evolution of the automaton by mapping an operating mode to a set of first order, discrete time difference equations $F_{DE} \in \mathcal{F}_{DE}$ and algebraic equations $F_{AE} \in \mathcal{F}_{AE}$. The difference and algebraic equations are defined over $\mathbf{x}_c$, $\mathbf{w}_c$, and $\mathbf{v}_c$.
- $T : \mathcal{X}_d \rightarrow 2^{\mathcal{P} \,\cup\, \mathcal{C}}$ specifies the evolution of the automaton's operating mode using probabilistic transitions between discrete assignments. T maps an operating mode to a finite set of transition probabilities $p_i \in \mathcal{P}$ and their associated guard conditions $c_i \in \mathcal{C}$. The guard conditions are defined over the continuous state variables and discrete control variables, and must completely partition $\mathbb{R}^{n_x} \times U_d$.

2.1.1 Example PHA - Electrical Load

We now model the electrical load with heat exchanger from the example fluid system as a PHA. It has two modes: on and off.

$$\mathcal{X}_d = \{\text{on}, \text{off}\}$$

The component's continuous state is defined by its temperature.

$$\mathbf{x}_c = \{T\}$$

It interfaces with other components through the pressure and flow of water at both ends. Additionally, it has a temperature sensor and two noise inputs from the environment.

$$\mathbf{w}_c = \{p_1, p_2, f_1, f_2, y_T\}$$

$$\mathbf{v}_c = \{\nu_t, \nu_o\}$$

The interface and state variables are related using a combination of algebraic and difference equations. In both modes, the flow through the load is conserved, the pressure drop across both ends is related to the flow rate and the resistance to the flow (a constant, R), and the temperature sensor reads the temperature of the component subject to Gaussian noise (ν_o).

In both operating modes, the temperature of the component changes relative to the difference in temperatures between the component and the water (T_W, assumed a constant), the exchanger's efficiency and water's heat capacity (bundled together in the constant k_1), the rate of flow through the exchanger, and the heat capacity of the component (the constant k_2). When the load is on, there is an additional contribution due to the heat (Q) being generated by the component. These equations are summarized in Table 1.

Table 1: **F** of load.

$\mathbf{x}_d$	**F**
on	$f_1 + f_2 = 0$ $p_1 - p_2 = R * f_1$ $y_T = T + \nu_o$ $T_{k+1} = T_k + k_2^{-1}(-k_1(T_k - T_W) * f_1 + Q) * \Delta t + \nu_t$
off	$f_1 + f_2 = 0$ $p_1 - p_2 = R * f_1$ $y_T = T + \nu_o$ $T_{k+1} = T_k + k_2^{-1}(-k_1(T_k - T_W) * f_1) * \Delta t + \nu_t$

Loads have one command variable that can be used to command the load to turn on or off.

$$U_d = \{\text{turn-on}, \text{turn-off}\}$$

Figure 3 shows the transition function for the automaton in graphical form. Guard condition c_1 is true if the command variable is set to "turn-off." Guard condition c_2 is true if the command variable is set to "turn-on." Guard condition c_3 is true if the command value is not "turn-off" and the temperature of the load is greater then 350K. If c_1 holds when the load is "on," the load will transition to the "off" state with 100% certainty. If c_3 holds when the load is on, the load will shut

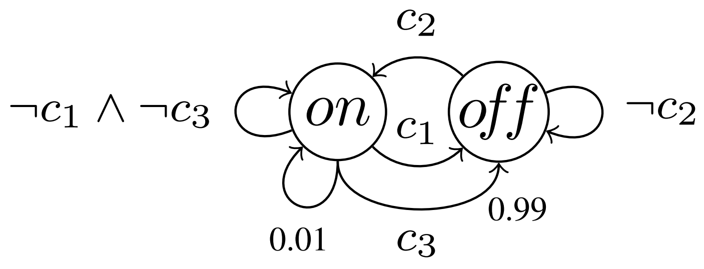


Figure 3: Load transition diagram.

off with high probability (representing a fail safe for the temperature being too high). If c_2 holds when the load is "off," the load will transition to the "on" state with 100% certainty, otherwise the load will stay "off."

2.2 Concurrent Probabilistic Hybrid Automata

At the system level, the artifact being monitored can be viewed as a composition of PHAs operating concurrently. While a composition of PHA models is also a PHA (Blackmore et al., 2008), we do not join the component PHAs together to a single PHA. This allows us to exploit the factored representation that keeping them separate allows. The formal definition of a CPHA is found in Definition 2. Intuitively, a CPHA is a set of component PHAs and information about the system level inputs and outputs.

Definition 2. *A concurrent probabilistic hybrid automaton (CPHA) model is described by the 3-tuple $\langle \mathbf{A}, \mathbf{u}, \mathbf{y_c} \rangle$, where:*

- $\mathbf{A}$ is a set of component PHAs.
- $\mathbf{u} = \{\mathbf{u}_d, \mathbf{u}_c\}$ is the set of system level input variables, i.e. the variables directly controllable by the operator of the entire system. $\mathbf{u}_d$ is the union of the command variables for every PHA in $\mathbf{A}$. $\mathbf{u}_c$ is a subset of the union of the continuous interface ($\mathbf{w}_c$) variables of every PHA in $\mathbf{A}$.
- $\mathbf{y}_c$ is the set of system level observable variables, i.e. the variables directly observable by the operator of the entire system. It is a subset of the union of the continuous interface ($\mathbf{w}_c$) variables of every PHA in $\mathbf{A}$. $\mathbf{y}_c$ and $\mathbf{u}_c$ must be disjoint.

In the example system, the CPHA consists of six separate PHAs: the two loads, two valves, one T-junction, and one pump. They are connected in the expected way, the pressure at each connection must be the same and the flows must be opposite (flow is defined to be positive into the component). Additionally, the pressures at the pump's inlet and the loads' outlets are constrained to be 0. The only observables from the system are the temperatures of the two loads.

The set of equations active when the pump and loads are on and the valves are open are:

$$\begin{aligned}
p_1 &= P && (1)\\
p_2 &= p_1 && (2)\\
p_3 &= p_1 && (3)\\
f_1 + f_2 + f_3 &= 0 && (4)\\
p_2 - p_4 &= Rf_2 && (5)\\
f_4 + f_2 &= 0 && (6)\\
p_3 - p_5 &= Rf_3 && (7)\\
f_5 + f_3 &= 0 && (8)\\
f_6 + f_4 &= 0 && (9)\\
p_4 - p_6 &= Rf_4 && (10)\\
y_1 &= T_1 + \nu_{o,1} && (11)\\
T_{1,k+1} &= T_{1,k} + k_2^{-1}(-k_1(T_{1,k} - T_W) * f_4 + Q) * \Delta t + \nu_{t,1} && (12)\\
p_6 &= 0 && (13)\\
f_7 + f_5 &= 0 && (14)\\
p_5 - p_7 &= Rf_5 && (15)\\
y_2 &= T_2 + \nu_{o,2} && (16)\\
T_{2,k+1} &= T_{2,k} + k_2^{-1}(-k_1(T_{2,k} - T_W) * f_5 + Q) * \Delta t + \nu_{t,2} && (17)\\
p_7 &= 0 && (18)
\end{aligned}$$

Given a mode assignment for the component PHAs, a symbolic solver can be used to solve for the next continuous state and outputs of the system using the set of algebraic and difference equations provided by F and the current continuous state and system level inputs. This results in equations of the form below, where Equation 19 represents the continuous state of the system and Equation 20 represents the continuous valued observations of the system.

$$\mathbf{x}_{c,k} = \mathbf{f}_k(\mathbf{x}_{c,k-1}, \mathbf{u}_{c,k}) + \mathbf{v}_{s,k} \quad (19)$$

$$\mathbf{y}_{c,k} = \mathbf{g}_k(\mathbf{x}_{c,k}, \mathbf{u}_{c,k}) + \mathbf{v}_{o,k} \quad (20)$$

The solved set of equations for the example system are provided by Equations 21 and 22.

$$\begin{pmatrix} T_{1,k} \\ T_{2,k} \end{pmatrix} = \begin{pmatrix} \left(\frac{-k_1(T_{1,k-1}-T_W)P}{k_2(R_v+R_l)} + \frac{Q}{k_2}\right)\Delta t \\ \left(\frac{-k_1(T_{2,k-1}-T_W)P}{k_2(R_v+R_l)} + \frac{Q}{k_2}\right)\Delta t \end{pmatrix} + \begin{pmatrix} \nu_{l_1} \\ \nu_{l_2} \end{pmatrix} \quad (21)$$

$$\begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \begin{pmatrix} T_{1,k} \\ T_{2,k} \end{pmatrix} \quad (22)$$

One interesting feature of PHAs and CPHAs is that causality relations between interface variables are not made explicit unless the variable is a system level input or output. In one mode a continuous I/O variable may be determined by a PHA, but in another mode it may be required to determine the state or other continuous I/O variables of the same PHA. However, given a mode assignment, causality can be determined using the system level inputs and outputs from the CPHA definition. This is accomplished using a combination of causal analysis (Nayak, 1995; Trave-Massuyes & Pons, 1997), structural analysis (Reinschke, 1988; Gehin, Assas, & Staroswiecki, 2000) and graph decomposition. See the previous work on HME (Hofbaur & Williams, 2004) for a more in-depth discussion of this feature and its application to *unknown* modes where the equations of an operating mode are not known.

3. Discrete Time, Hybrid State Estimation for CPHAs

In this section, we discuss the theory underlying discrete time, hybrid state estimators for CPHA models. The goal of CPHA estimators is to compute the *belief state* of a CPHA, given the system level inputs and observations from the previous time steps. The belief state is presented in Equation 23, where $\mathbf{m}_k$ represents the mode (discrete state) of the system at time k, $\mathbf{x}_{c,k}$ represents the continuous state of the system at time k, $\mathbf{y}_{1:k}$ represents all observations of the system up to and including time k, and $\mathbf{u}_{1:k}$ represents all commands to the system up to and including time k.

$$p(\mathbf{m}_k, \mathbf{x}_{c,k} | \mathbf{y}_{1:k}, \mathbf{u}_{1:k}) \tag{23}$$

3.1 Belief State Representation

The hybrid state estimator described in this paper represents the belief state as a mixture of constituent probability distributions. If there are N non-zero components to the mixture, Equation 24 describes how the complete belief state is obtained from the individual mixture components.

$$p(\mathbf{m}_k, \mathbf{x}_{c,k} | \mathbf{y}_{1:k}, \mathbf{u}_{1:k}) = \sum_{0 \le j < N} b(\mathbf{m}_k, \mathbf{x}_{c,k}; \hat{\mathbf{x}}_k^{(j)}) \tag{24}$$

Where b represents the belief state of the mixture component parameterized by $\hat{\mathbf{x}}_k^{(j)}$. Each mixture component is parameterized by a weight $w_k^{(j)}$, mode assignment $\hat{\mathbf{m}}_k^{(j)}$, and continuous probability distribution $p_{c,k}^{(j)}$:

$$\hat{\mathbf{x}}_k^{(j)} = \left\langle w_k^{(j)}, \hat{\mathbf{m}}_k^{(j)}, p_{c,k}^{(j)} \right\rangle \tag{25}$$

$$b(\mathbf{m}_k, \mathbf{x}_{c,k}; \hat{\mathbf{x}}_k^{(j)}) = w_k^{(j)} \cdot p_{c,k}^{(j)}(\mathbf{x}_{c,k} | \hat{\mathbf{m}}_k^{(j)}, \mathbf{y}_{1:k}^{(j)}, \mathbf{u}_{1:k}^{(j)}) \cdot \left[\mathbf{m}_k = \hat{\mathbf{m}}_k^{(j)}\right] \tag{26}$$

Note in Equation that b is nonzero only if the input mode matches the mode in the parameterization.

3.2 Hybrid State Estimation

Figure 4 shows the relationship between the mode, continuous state, commands, and observations of a CPHA as a dynamic Bayes net. The mode at a time step is dependent only on the discrete command at the current time as well as the mode and continuous state at the previous time step. The continuous state at the current time step is dependent on only the current continuous command and mode as well as the previous continuous state. Last, the observations are dependent only on the current mode, continuous state, and continuous command. These dependencies follow directly from the prior PHA and CPHA definitions.

Given these dependencies, a standard recursive approach for Bayesian state estimation, modified to account for the existence of both discrete and continuous state variables, can be used. The

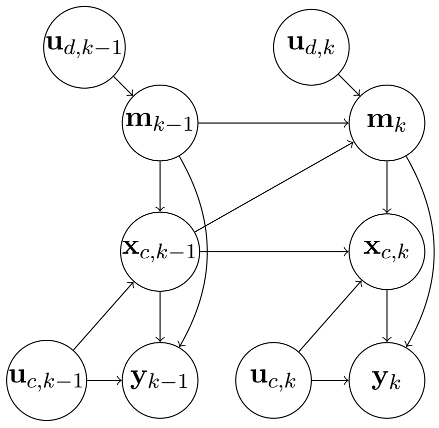


Figure 4: The recursive hybrid estimation problem shown as a dynamic Bayes net.

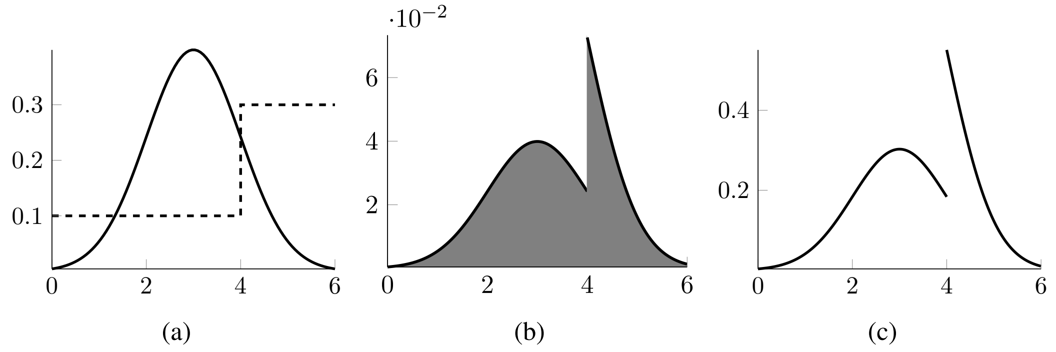


Figure 5: Interaction between continuous state and mode transitions. In (a), the solid line represents the continuous belief at the previous time $(p_{c,k-1})$ and the dashed line represents the probability of transitioning to the target mode as a function of the continuous state (τ). (b) shows the unnormalized multiplication of $p_{c,k-1}$ and τ. The area under the curve represents the probability of transitioning to the target mode given the previous continuous belief. (c) shows the properly normalized prior to be used by the continuous state estimator.

remainder of this section discusses how to compute a new component distribution at time k, starting from a component distribution $\hat{\mathbf{x}}_{k-1}^{(i)}$, and assuming a target mode assignment $\hat{\mathbf{m}}_k^{(j)}$ is provided.

3.2.1 Continuous State Estimation

As the target mode is known, the state dynamics and observation functions of the system are known and the system wide dynamics equations can be computed in the forms shown in Equations 19 and 20. Additionally, the probability of transitioning to the target mode can be computed as a function of the continuous state (represented as $\tau(\mathbf{x}_c)$).

Before a continuous state estimator can be invoked with the computed dynamics, the prior belief for the continuous state estimator must be computed. The prior is not simply $p_{c,k-1}$ due to the coupling between continuous state and mode transition probabilities. For instance, a component might fail only if its temperature is above a specified threshold, so any probability mass below that threshold should not be considered when the target mode is the failed mode.

Instead, the initial continuous belief (p_0) for the continuous estimator is the previous belief, weighted by τ, as shown in Equation 27. Figure 5 also illustrates this concept. Note how there can be discontinuities in the prior for any system where the mode transition is dependent on continuous state.

$$p_0(\mathbf{x}_c) \propto p_{c,k-1}(\mathbf{x}_c) \cdot \tau(\mathbf{x}_c) \tag{27}$$

Some prior work, such as HME (Hofbaur & Williams, 2002, 2004) has ignored this subtlety and used $p_{c,k-1}$ as the prior. This allows for standard Kalman filter variants such as the Extended Kalman Filter (EKF) (Sorenson, 1985) and Unscented Kalman Filter (UKF) (Julier & Uhlmann, 1997) to be used as the prior is always Gaussian. Other prior work has used particle filters (Blackmore et al., 2008) which do not require the prior to be a normal distribution.

Another approach, used by this work in the experimental section, is to use a Kalman filter variant that allows these constraints to be expressed directly, such as the truncated Kalman filter varieties (e.g., the truncated unscented Kalman filter) (Teixeira, Tôrres, Aguirre, & Bernstein, 2010; Simon, 2010; Garcia-Fernandez, Morelande, & Grajal, 2012). In this case, the prior is broken into regions where τ is piecewise constant, the truncated Gaussian in each section is projected to a non-truncated Gaussian with the same mean and variance, and that is used as the prior. This approach results in a single target mode having several children component distributions, each with a different continuous belief.

Without loss of generality, we assume that an extended Kalman filter is used to estimate the continuous state for the remainder of the technical discussion. The EKF represents the continuous belief as a Gaussian distribution by tracking the belief's mean ($\hat{\mathbf{x}}_{c,k|k}$ is the mean at time k using information available up to time k) and covariance ($\mathbf{P}_{k|k}$ is the covariance of the continuous belief at time k using information available up to time k). The following equations summarize the process of producing a new continuous belief state for time k using the belief from time $k-1$. $\mathbf{f}$ and $\mathbf{g}$ are the functions described in Equations 19 and 20 respectively and $\mathbf{z}_k$ is the observed values of the CPHA continuous outputs.

$$\mathbf{A}_{k-1} = \left.\frac{\partial \mathbf{f}_k}{\partial \mathbf{x}_c}\right|_{\hat{\mathbf{x}}_{c,k-1|k-1},\mathbf{u}_{c,k}} \tag{28}$$

$$\mathbf{C}_{k} = \left.\frac{\partial \mathbf{g}_k}{\partial \mathbf{x}_c}\right|_{\hat{\mathbf{x}}_{c,k|k-1},\mathbf{u}_{c,k}} \tag{29}$$

$$\hat{\mathbf{x}}_{c,k|k-1} = \mathbf{f}_k(\hat{\mathbf{x}}_{c,k-1|k-1}, \mathbf{u}_{c,k}) \tag{30}$$

$$\mathbf{P}_{k|k-1} = \mathbf{A}_{k-1}\mathbf{P}_{k-1|k-1}\mathbf{A}_{k-1}^{\top} + \mathbf{Q}_{k-1} \tag{31}$$

$$\widetilde{\mathbf{y}}_k = \mathbf{z}_k - \mathbf{g}_k(\hat{\mathbf{x}}_{c,k|k-1}, \mathbf{u}_{c,k}) \tag{32}$$

$$\mathbf{S}_k = \mathbf{C}_k\mathbf{P}_{k|k-1}\mathbf{C}_k^{\top} + \mathbf{R}_k \tag{33}$$

$$\mathbf{K}_k = \mathbf{P}_{k|k-1}\mathbf{C}_k^{\top}\mathbf{S}_k^{-1} \tag{34}$$

$$\hat{\mathbf{x}}_{c,k|k} = \hat{\mathbf{x}}_{c,k|k-1} + \mathbf{K}_k\widetilde{\mathbf{y}}_k \tag{35}$$

$$\mathbf{P}_{k|k} = (\mathbf{I} - \mathbf{K}_k\mathbf{C}_k)\mathbf{P}_{k|k-1} \tag{36}$$

3.2.2 WEIGHT CALCULATION

The weight of the target component distribution is computed using standard discrete Bayesian techniques. First, the weight is computed using only the transition probability, ignoring the observations, as shown in Equation 37. Then the observations are assimilated and the weight updated as shown in Equation 38. The hybrid transition function (P_T) and hybrid observation function (P_O) are described in the following subsections.

$$w_{k|k-1}^{(j)} = P_T(\hat{\mathbf{m}}_k^{(j)}, \hat{\mathbf{x}}_{k-1}^{(i)}, \mathbf{u}_{d,k}) w_{k-1}^{(i)} \tag{37}$$

$$w_k^{(j)} = \frac{P_O(\mathbf{y}_k, p_{c,k}^{(j)}, \mathbf{u}_{c,k}) w_{k|k-1}^{(j)}}{\sum_i P_O(\mathbf{y}_k, p_{c,k}^{(i)}, \mathbf{u}_{c,k}) w_{k|k-1}^{(i)}} \tag{38}$$

3.2.3 HYBRID TRANSITION FUNCTION

The hybrid transition function, P_T is defined as the probability of changing into a mode, given both the previous mode and previous continuous belief. It is computed by taking the integral of the source continuous belief, weighted by τ, as described by Equation 39. This probability is also shown in Figure 5b.

$$P_T(\hat{\mathbf{m}}_k^{(j)}, \hat{\mathbf{x}}_{k-1}^{(i)}, \mathbf{u}_{d,k}) \triangleq \int_{\mathbf{x}_c} p_{c,k-1}(\mathbf{x}_c)\tau(\mathbf{x}_c) \tag{39}$$

3.2.4 HYBRID OBSERVATION FUNCTION

The hybrid observation function, P_O is defined as the probability of receiving the observed outputs of the system, as defined by Equation 40

$$P_O(\mathbf{y}_{c,k}, \hat{\mathbf{x}}_{k-1}^{(i)}, \mathbf{u}_{c,k}) \triangleq p(\mathbf{y}_{c,k} | \hat{\mathbf{m}}_k^{(j)}, \hat{\mathbf{x}}_{k-1}^{(i)}, \mathbf{u}_{c,k}) \tag{40}$$

If a Kalman filter variant is used as the continuous estimator, P_O is determined using the residual of the expected measurements and the computed covariance of this residual. For an EKF, the residual is computed using Equation 32 and the covariance of the residual is computed using Equation 33. The probability of this observation is then computed as in Equation 41[1].

$$P_O(\mathbf{y}_{c,k}, \hat{\mathbf{x}}_{k|k-1}^{(j)}, \mathbf{u}_{c,k}) = \frac{1}{|2\pi \mathbf{S}_k^{(j)}|^{1/2}} e^{-0.5(\tilde{\mathbf{y}}_k^{(j)})^\top (\mathbf{S}_k^{(j)})^{-1} \tilde{\mathbf{y}}_k^{(j)}} \tag{41}$$

While this is the strictly correct form of P_O for an EKF, Maybeck and Stevens (1991) suggest dropping the normalization constant as it can bias the estimate to incorrect mode estimates, particularly if some modes represent sensor failures and the determinant of $\mathbf{S}$ is significantly different in those modes. An additional benefit to ignoring this normalization constant is it guarantees that $0 \leq P_O \leq 1$, a property that the best first estimator will exploit. This results in the hybrid observation function shown in Equation 42.

$$P_O(\mathbf{y}_{c,k}, \hat{\mathbf{x}}_{k|k-1}^{(j)}, \mathbf{u}_{c,k}) = e^{-0.5(\tilde{\mathbf{y}}_k^{(j)})^\top (\mathbf{S}_k^{(j)})^{-1} \tilde{\mathbf{y}}_k^{(j)}} \tag{42}$$

1. As the observations are continuous, $p(\mathbf{y}_{c,k}| \ldots)$ is actually a probability density function and would need to be multiplied by some $d\mathbf{y}$ to obtain a true probability. However, this term is ignored because it has no impact after the normalization shown in Equation 38 occurs.

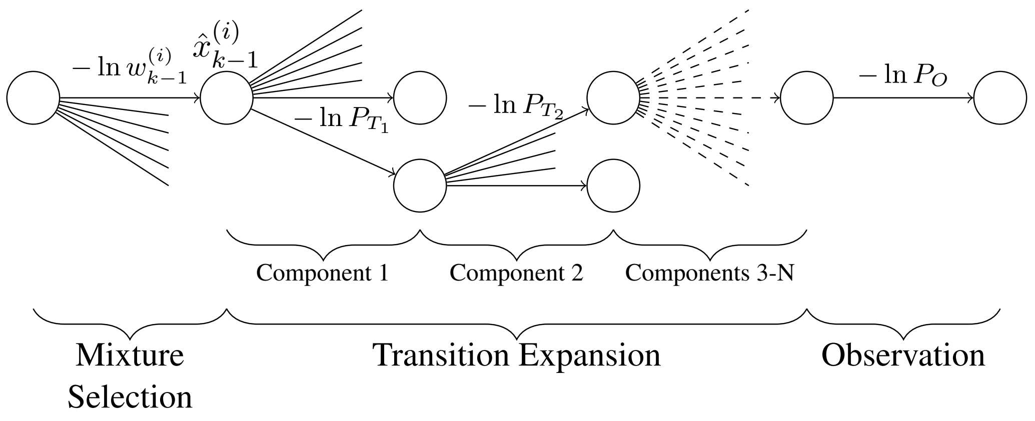


Figure 6: Search tree for hybrid state estimation.

3.3 The Need for Approximate State Estimation

Thus far, we have been assuming a target mode assignment is given. We now look at how target modes are chosen. A trivial method to compute the full belief state at time k is to enumerate each reachable mode for each component distribution at time $k-1$. However, this quickly becomes intractable, as the number of mode trajectories is exponential in both the number of component automata in the system and the number of time steps that have occurred since the start of estimation. In order to handle this intractability, two broad classes of methods have emerged: merging and pruning.

Merging methods seek to collapse component distributions together to reduce the number of number of component distributions being tracked. Methods using this technique include IMM (Blom & Bar-Shalom, 1988), GPB (Ackerson & Fu, 1970), detection-estimation methods (Tugnait, 1982), residual Kalman filter banks (Hanlon & Maybeck, 2000) and adaptive MM methods (Li & Bar-Shalom, 1996; Li et al., 1999; Li, 2000). These techniques generally require that at least one component distribution be kept for every possible mode of the system. For example, IMM methods use a Kalman filter for each mode of the system, and mix together all of the continuous distributions ending in a single mode into one normal distribution. While merging methods generally remove the time dependency for the number of distributions computed, they still grow exponentially in the number of component automata in the system.

Pruning deals with the intractability by maintaining only a likely subset of the possible component distributions. Any pruned trajectory has a weight of zero. Within pruning methods there are two general approaches: stochastic sampling and k-best enumeration[2]. Additionally, previous work by Blackmore, et al. (2008) has shown that k-best and stochastic methods can be combined, resulting in an algorithm containing the strengths of both. This paper focuses on improvements to k-best estimation.

k-best state estimation can be accomplished by framing the state estimation problem as a best-first search problem. Figure 6 shows the search space for a fixed order of assignment to the search variables. The first set of branches represents choosing a hypothesis component distribution from the set at time $k-1$. The cost of that step is the negative log of the weight of that component. The

2. When used as a sub script, k indicates a time stamp. In all other cases it should be clear if k is referring to a time stamp or the number of tracked trajectories.

next N sets of branches represent choosing a transition for a single component in the CPHA. The cost for each of these steps is the negative log of the probability of that transition occurring given the chosen trajectory hypothesis. For the last step of the search there is only one branch. The cost for taking this branch is the negative log of the value of the hybrid observation function.

The total reward for a complete trajectory hypothesis at time k is then computed by multiplying the individual rewards. The weight of a trajectory hypothesis j is then computed by normalizing the rewards for every complete hypothesis. This results in an any-time hybrid state estimation algorithm that greedily generates the most likely trajectories at any time step.

4. Best-first Enumeration using Bounding Conflicts

The previous section motivated the need for an approximate state estimator and provided a sketch of how a k-best hybrid state estimator could be built using a best-first enumeration algorithm. In this section, we describe the A* with Bounding Conflicts (A*BC) algorithm, a bounding conflict-directed best-first enumeration algorithm.

The A* with Bounding Conflicts algorithm builds upon the Constraint-based A* and Conflict-directed A* algorithms (Williams & Ragno, 2007). A*BC searches through the space of partial assignments to a set of variables. Each member of the search space has as its neighbors the partial and full assignments realized by adding one more assignment to its partial assignment. The goals in the search space are every full assignment to the variables.

Definition 3. *An A*BC search problem is described by the tuple* $\langle Y, D, c, f \rangle, where:$

- $Y = \{y_1, \ldots, y_m\}$ is a set of m *decision variables* with finite domains $D = \{d_1, \ldots, d_m\}$.
- $c : Y \to \mathbb{R}$ is the cost function. c maps full assignments to the decision variables to a real valued cost.
- $f : 2^Y \to \mathbb{R}$ is the bounding function. f maps partial assignments to the decision variables to a lower bound on the cost of any full assignment containing that partial assignment.

The decision variables in the hybrid estimation enumeration problem correspond to the component distribution at time $k-1$ and the modes for each component at time k.

The bounding function f is the maximum likelihood of any full assignment that contains the current partial assignment. For the hybrid estimation enumeration problem, f is simply a function that computes the probability of the transitions already committed to, assumes the uncommitted PHAs take their most likely transition, and that the predicted observations match the actual observations exactly.

The cost function c, shown in Equation 43 is the probability of the system following the chosen transitions and the true likelihood of the observations being seen. This requires a Kalman filter to be run in order to calculate the hybrid observation function.

$$c = -\ln w_{k-1}^{(i)} - \ln P_T(\hat{\mathbf{m}}_k^{(j)}, \hat{\mathbf{x}}_{k-1}^{(i)}, \mathbf{u}_k) - \ln P_O(\mathbf{y}_{c,k}, \hat{\mathbf{x}}_{k-1}^{(i)}, \mathbf{u}_{c,k}) \tag{43}$$

4.1 A*BC in a Nutshell

A*BC is an optimal, best first search and enumeration algorithm. It is designed for problems where a weak bounding function is known at the beginning of the search, but there are opportunities during

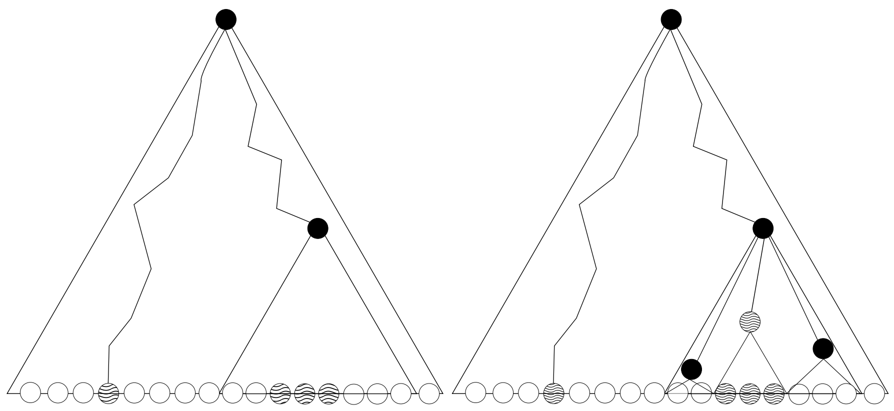


Figure 7: In the left figure, a bounding conflict has been learned that affects the shaded explored and unexplored nodes. In the figure on the right, the node is expanded such that one child covers the bounding conflict and the other children cover the remainder of the space.

the search to learn tighter bounding functions that apply to some subset of the states. A*BC incorporates these tighter bounding functions as they become known by updating the cost function and changing the order in which neighboring states are explored. In the case of hybrid state estimation, the bounding function is weak because it assumes that the predicted observations exactly match the actual observations.

The information on the bounding function gathered during the search is represented using *bounding conflicts*. A bounding conflict compactly represents a region of the search space and a tighter bounding function that is valid for that region. The learned bounding conflicts are then used during the search to proactively avoid exploring the regions covered by bounding conflicts until absolutely necessary.

Specifically, when A*BC expands a node to generate a set of children, it checks the node to be expanded against the current set of known bounding conflicts. If the set of full assignments represented by the search node intersects with the candidates denoted by a bounding conflict, one child of the node is generated to represent the intersection and the remaining children are generated to represent the remaining candidates. The bound of the child representing the intersection is then calculated using the tighter bounding function provided by the bounding conflict. This pushes that child deeper into the search queue, which lets the search algorithm delay exploring that region of the search space when compared to using only the original bounding function. This is illustrated by Figure 7.

4.2 Bounding Conflicts

The key innovation of A*BC is the use of *bounding conflicts* to compactly represent areas of the search space where the original bounding function is weak. Formally:

Definition 4. A bounding conflict is a pair $\langle z, b\rangle$ where:

- z is a partial assignment to the decision variables of an informed search problem and
- $b : P_z \rightarrow \mathbb{R}$ is a function that maps partial assignments that are extensions of z to a bound on the cost for any extension to z.

Any extension to the partial assignment of a bounding conflict is said to "manifest" the bounding conflict. Any partial assignment that entails the negation of the bounding conflict is said to "resolve" the conflict.

From the above definition of bounding conflicts, we can see that a classical conflict is simply a specialization of a bounding conflict. A classical conflict defines a region of infeasibility. Infeasibility can be represented as an infinitely high cost, therefore the tighter bound for a classical conflict is $b(x) = \infty$.

4.3 Search Algorithm

In this subsection we describe the A*BC algorithm in depth.

A node in the search tree is defined by the pair $\langle s, \beta \rangle$. s is a partial assignment to the problem's decision variables. β is the set of bounding conflicts that are manifested by the node's partial assignment.

In the described search space if there are m variables, each full assignment has $m!$ distinct, equal cost paths leading to it that differ only in the order in which variables are assigned. However, A*BC only considers one path to each full assignment. It accomplishes this by generating only a subset of the neighbors for each search node. The sets of full assignments reachable from the generated neighbors are guaranteed to be exhaustive and mutually exclusive. This principle of following only a single path to each full assignment has been previously used by Constraint-Based A* and Conflict-Directed A*.

The A*BC algorithm is described by Algorithm 1. It is conceptually similar to A*. Nodes are stored on an open queue, sorted by the bound function f. Once a node is popped off (lines 5-6), it is first checked to see if it represents a full assignment. If it is, it is added to the list of solutions and the next node is popped off (lines 7-8).

If the node does not represent a full assignment, its neighbors are generated and then queued (lines 10-23). The expansion process starts by first determining which bounding conflicts the node manifests (Γ_m) and which bounding conflicts the node neither manifests nor resolves (Γ_n). The node's set of manifested bounding conflicts is then updated appropriately (lines 10-11).

After the node is updated, its neighbors are generated by one of two methods. First, if a node has no bounding conflicts it neither manifests nor resolves (Γ_n is empty), the neighbors are generated by the NEIGHBORS-BY-VARIABLE method. Otherwise, a bounding conflict from Γ_n is chosen and used to generate the neighbors with the NEIGHBORS-BY-BC method.

Once the neighbors are generated, they are then checked to determine if they full assignments. If they are and the bound calculated by the bounding function is sufficiently far from the true cost, then a bounding conflict is learned to learn why. Both BOUNDS-INADEQUATE and LEARN-BC are application specific and allow some tuning (lines 18-22).

Neighbors by Variable The NEIGHBORS-BY-VARIABLE algorithm has been omitted for space. It works by first choosing an unassigned variable and generating search nodes for every assignment to the variable. The neighbors simply inherit the set of manifested bounding conflicts from the node being expanded. This is the same method of neighbor generation used by constraint-based A*.

Neighbors by Bounding Conflict When a node neither resolves nor manifests a bounding conflict, the NEIGHBORS-BY-BC method is used to generate its neighbors, described in Algorithm 2. Given a node and a bounding conflict, this method generates neighbor nodes by creating one node

```
1  open ← {MAKE-NODE({}, {})};
2  solutions ← {};
3  $\Gamma \leftarrow \{\}$;
4  while not TERMINATE?() do
5      $n \leftarrow \mathrm{argmax}_{n_i \in \text{open}} f(n_i)$;
6      open ← open − $n$;
7      if FULL-ASSIGNMENT?($n$) then
8          solutions ← solutions ∪ {$n$};
9      else
10         $\Gamma_n, \Gamma_m$ ← FIND-RELEVANT-CONFLICTS($n, \Gamma$);
11         $n[\beta] \leftarrow n[\beta] \cup \Gamma_m$;
12         if $\Gamma_n$ not empty then
13             $\gamma$ ← an element of $\Gamma_n$;
14             neighbors ← NEIGHBORS-BY-BC($n, \gamma$);
15         else
16             neighbors ← NEIGHBORS-BY-VARIABLE($n$);
17         end
18         foreach neighbor ∈ neighbors do
19             if FULL-ASSIGNMENT?(neighbor) and BOUNDS-INADEQUATE?(neighbor)
               then
20                 $\gamma$ ← LEARN-BC(neighbor);
21                 $\Gamma \leftarrow \Gamma \cup \{\gamma\}$;
22             end
23             open ← open ∪ {neighbor};
24         end
25     end
26 end
27 return solutions
```

Algorithm 1: A* with Bounding Conflicts

that manifests the conflict and then generating the remaining nodes to resolve the conflict and cover the remaining full assignments.

The bounding conflict is resolved by extending the node's partial assignment to a partial assignment that implies the negation of the bounding conflict. To avoid generating multiple paths to the same partial assignment, extra care is taken to ensure that the full assignments reachable from each neighbor are disjoint. First, a variable assignment from the bounding conflict is chosen (line 3). Next, neighbors are generated for every value of that variable, except for the value involved in the bounding conflict (lines 4-8). Then, another assignment is chosen and the process repeated, except every new neighbor also includes the previously chosen assignments from the bounding conflict — ensuring that the full assignments reachable from each neighbor never overlap.

```
   Input: The node to exapnd, $n$.
   Input: The bounding conflict to use, $\gamma$.
   Output: The neighbors of $n$.
1  children ← MAKE-NODE($n[s] \cup \gamma[z], n[\beta] \cup \gamma$);
2  $\phi \leftarrow \{\}$;
3  foreach assignment $\in \gamma[z]$ do
4      $x \leftarrow$ variable of assignment;
5      $y \leftarrow$ value of assignment;
6      foreach $v \in (dom(x) - y)$ do
7          children ← MAKE-NODE($n[s] \cup \{x = v\} \cup \phi, n[\beta]$);
8      end
9      $\phi \leftarrow \phi \cup \{x = y\}$;
10 end
11 return children;
```

Algorithm 2: NEIGHBORS-BY-BC

4.4 Learning Bounding Conflicts

A bounding conflict describes a "gap" between the true and estimated costs of the best extension to a partial assignment. It is represented as a tuple $\langle z, b_z \rangle$ where b_z is a bounding function on partial assignments and z is a partial assignment where b_z is valid. This section describes how both components of a bounding conflict are learned for the hybrid state estimation problem in this work.

Recall that the cost function used during the search is (simplified):

$$c = -\ln w - \ln P_T - \ln P_O \tag{44}$$

The most difficult piece of this cost function to compute a tight bound for is the last — the sensor likelihood term — as it requires a continuous filter to be run. As such, the bounding function assumes the sensor readings perfectly match with the predictions so that $\ln P_O = 0$. However, once the Kalman filter used in this work is run, $\ln P_O$ can be decomposed into additive factors associated with subsets of the mode assignments. This is shown in Equation 46 using conditional probabilities. Additionally, the mode assignments each factor is conditioned on are reduced by using causal analysis (Nayak, 1995; Trave-Massuyes & Pons, 1997) to determine which modes introduce equations that the calculation of each factor's observation depends on.

$$\ln P_O(\mathbf{y}_{c,k}, \hat{\mathbf{x}}_{k-1}^{(i)}) \triangleq \ln p(\mathbf{y}_{c,k} | \hat{\mathbf{m}}_k^{(j)}, \hat{\mathbf{x}}_{k-1}^{(i)}) \tag{45}$$

$$\begin{aligned} \ln p(\mathbf{y}_{c,k} | \hat{\mathbf{m}}_k^{(j)}, \hat{\mathbf{x}}_{k-1}^{(i)}) &= \ln p(y_{1,c,k} | \hat{\mathbf{m}}_k^{(j)}, \hat{\mathbf{x}}_{k-1}^{(i)}) \\ &+ \ln p(y_{2,c,k} | y_{1,c,k}, \hat{\mathbf{m}}_k^{(j)}, \hat{\mathbf{x}}_{k-1}^{(i)}) \\ &+ \dots \\ &+ \ln p(y_{N,c,k} | \mathbf{y}_{1:N-1,c,k}, \hat{\mathbf{m}}_k^{(j)}, \hat{\mathbf{x}}_{k-1}^{(i)}) \end{aligned} \tag{46}$$

With this knowledge that the hybrid observation function is factorizable, learning a bounding conflict becomes equivalent to searching for a factorization where the first term (that is independent of all other observations) is as negative as possible. Bounding conflicts can be learned by using any subset of n observations. Of particular interest is learning bounding conflicts from a single observation, where the search can be done in time linear to the number of system-level output variables. This is the method used in the this work.

5. Example Execution

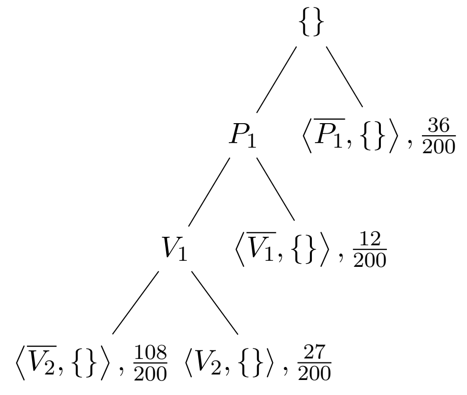
In this section, we provide an example execution of the state estimation algorithm, with A*BC on the running example. For compactness, we will specify that the pump is on by writing P_1 and that the pump is off by writing $\overline{P_1}$. Similarly for the valves, V_1 means the valve is open and $\overline{V_1}$ means the valve is closed. For brevity, we will also assume the loads are known to be constantly off. For simplicity and clarity, we will work with probabilities instead of negative $\log$ probabilities (as such, the goal is to maximize instead of minimize).

The initial discrete state of the system is known to be $\{P_1, V_1, V_2\}$ with complete certainty and the temperature of each load is described by the Gaussian distribution $\mathcal{N}(400, 1)$. After one time step, the temperature of each load is measured to be 398. Additionally, valve V_2 was sent the close command while valve V_1 and pump P_1 were sent commands to stay open and on, respectively. The transition probabilities given this situation are described below in Table 2.

Table 2: Probability of each component transitioning to a given mode.

Component	on/open	closed/off
P_1	$\frac{3}{4}$	$\frac{1}{4}$
V_1	$\frac{9}{10}$	$\frac{1}{10}$
V_2	$\frac{1}{5}$	$\frac{4}{5}$

The search begins with the root node, representing the empty partial assignment. No bounding conflicts are known at the start, so the SPLIT-BY-VARIABLE method is used. Assuming that A*BC chooses variables in the order P_1, V_1, then V_2, the state of the search when the first complete assignments are generated is shown in Figure 8.



$\{\}$
P_1
$\langle \overline{P_1}, \{\} \rangle, \frac{36}{200}$
V_1
$\langle \overline{V_1}, \{\} \rangle, \frac{12}{200}$
$\langle \overline{V_2}, \{\} \rangle, \frac{108}{200}$
$\langle V_2, \{\} \rangle, \frac{27}{200}$

Figure 8: State of the search when the first full assignments are generated. The probabilities according to the bounding function are shown for each node on the fringe. No bounding conflicts are known.

Next, the costs for the full assignments are computed. Figure 9 shows the update probabilities for the full assignments. Note that the probability of $\{P_1, V_1, \overline{V_2}\}$ has dropped by a factor of 6. This is because the Kalman filter predicts that, because there is no flow through load 2 (due to valve 2 being closed), the temperature in load 2 should stay at 400. However, the sensor has given a reading of 398. This is within the noise range of the sensor, but the likelihood of the sensor being that far off in this situation is determined to be $\frac{1}{6}$. However, the temperature of 398 is the most likely reading if valve 2 were open, so the probability of $\{P_1, V_1, V_2\}$ stays the same.

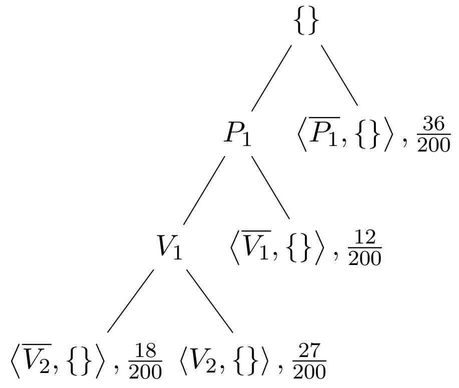

$\{\}$
P_1
$\left\langle \overline{P_1}, \{\} \right\rangle, \frac{36}{200}$
V_1
$\left\langle \overline{V_1}, \{\} \right\rangle, \frac{12}{200}$
$\left\langle \overline{V_2}, \{\} \right\rangle, \frac{18}{200}$
$\left\langle V_2, \{\} \right\rangle, \frac{27}{200}$

Figure 9: State of the search when the likelihoods of the full assignments are calculated using a Kalman filter.

At this point, a bounding conflict can be learned. The bound produced by f for any solution containing $\{\overline{V_2}\}$ will always be higher than the true likelihood by a factor of at least 6. This results in the bounding conflict $\gamma_1 = \left\langle \{\overline{V_2}\}, f(x) * \frac{1}{6} \right\rangle$.

Now, A*BC dequeues the next best node: $\{\overline{P_1}\}$. This partial solution does not resolve or manifest γ_1, so the NEIGHBORS-BY-BC method is used to generate its neighbors. The first neighbor manifests γ_1 by extending $\{\overline{P_1}\}$ to include $\{\overline{V_2}\}$. Additionally, this node is annotated with the fact that it manifests γ_1, which allows its bound to be computed using $\gamma_1[b]$. The second neighbor extends the node to include the negation of the bounding conflict, $\{V_2\}$. The state of the search at this point is shown in Figure 10.

Now, the next best node is $\{P_1, V_1, V_2\}$. If bounding conflicts were not used, the full assignment $\{\overline{P_1}, V_1, \overline{V_2}\}$ would had to be generated first.

6. Results

To demonstrate the effectiveness of the A*BC algorithm as a best-first enumerator for hybrid state estimation, it was implemented and compared to HME without bounding conflicts on two examples. The first is a reproduction of the three PHA system from Hofbaur (2004). The second is the shipboard fluid system analog used as this work's motivation.

The experiments were implemented in portable Common Lisp and performed using Steel Bank Common Lisp (SBCL) on a single core of an Intel Core i7-7700HQ processor. The Weyl computer algebra system (Zippel, 1993) was used as the symbolic solver used to solve the algebraic and difference equations into the form shown in Equations 19 and 20. The large benchmark (shipboard cooling system) requires the solving of hundreds of algebraic and difference equations, which was

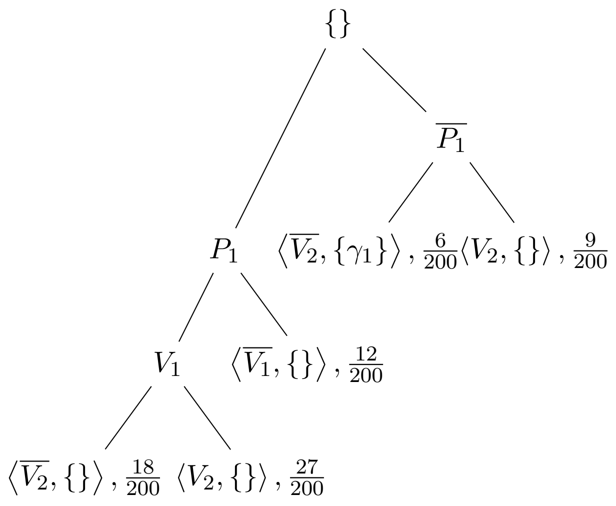

$\{\}$
$\overline{P_1}$
P_1
$\langle \overline{V_2}, \{\gamma_1\} \rangle, \frac{6}{200}$
$\langle V_2, \{\} \rangle, \frac{9}{200}$
V_1
$\langle \overline{V_1}, \{\} \rangle, \frac{12}{200}$
$\langle \overline{V_2}, \{\} \rangle, \frac{18}{200}$
$\langle V_2, \{\} \rangle, \frac{27}{200}$

Figure 10: State of the search after the first neighbors are generated by NEIGHBORS-BY-BC.

observed to stress Weyl. As improving Weyl was not in scope for this research, all results for the large benchmark use a timeout of one hour and use the number of Kalman filter executions as a proxy for time.

6.1 Simple Automaton

This example comes originally from Hofbaur and Williams (2004). It consists of three automaton with a total of three state variables, one continuous input, and two continuous outputs. The difference and algebraic equations for each PHA are described in Tables 3, 4, and 5. The transition diagrams for the PHAs are show in Figure 11

Table 3: Algebraic and difference equations for PHA 1.

Mode	Equations
m_{11}	$u_{c1} = 2.0w_{c1}$
m_{12}	$u_{c1} = -2.0w_{c1}$

Table 4: Algebraic and difference equations for PHA 2.

Mode	Equations
m_{21}	$x_{c1,k+1} = 0.95x_{c1,k} + w_{c1}$ $y_{c1} = 2.0x_{c1}$
m_{22}	$x_{c1,k+1} = 0.6x_{c1,k} + w_{c1}$ $y_{c1} = 2.0x_{c1}$
m_{23}	$x_{c1,k+1} = 1.01x_{c1,k} + w_{c1}$ $y_{c1} = 2.0x_{c1}$

The previous work has already shown that a k-best enumeration style hybrid estimator with A* as the search algorithm is faster than a standard IMM style approach up to $k = 20$ on this problem.

Table 5: Algebraic and difference equations for PHA 3.

Mode	Equations
m_{31}	$x_{c2,k+1} = x_{c3,k} + 0.2y_{c1}$ $x_{c3,k+1} = -0.63x_{c2,k} + 1.6x_{c3,k} + 0.1u_{c1}$ $y_{c2} = 0.5x_{c2} + 0.1x_{c3}$
m_{32}	$x_{c2,k+1} = x_{c3,k} + 0.2y_{c1}$ $x_{c3,k+1} = -0.8x_{c2,k} + 1.6x_{c3,k} + 0.1u_{c1}$ $y_{c2} = 0.5x_{c2} + 0.1x_{c3}$
m_{33}	$x_{c2,k+1} = x_{c3,k} + 0.2y_{c1}$ $x_{c3,k+1} = -0.3x_{c2,k} + 1.1x_{c3,k} + 0.1u_{c1}$ $y_{c2} = 0.5x_{c2} + 0.1x_{c3}$

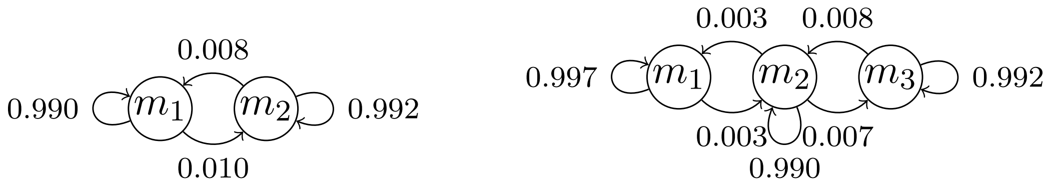


Figure 11: Simple concurrent automata transitions. The left diagram shows the transition probabilities for PHA 1. The right diagram shows the transition probabilities for PHAs 2 and 3.

Additionally, the relative error is worse than IMM, but comparable (0.1130 for an IMM approach and 0.1172 for $k = 20$).

The results of comparing a k-best hybrid estimator with A*BC vs. standard A* for 500 timesteps of the three PHA system are shown in Table 6. The k-best runtime column shows the normalized time needed to process all timesteps with $k = 20$ and A* search as the comparison standard. The IMM runtime column restates the original data collected by Hofbaur and Williams comparing the k-best estimator to a standard IMM approach. The middle set of columns records the maximum and average number of times a complete Kalman filter needed to be run for each timestep, along with the total fraction of time spent in the Kalman filtering routine (including both derivation and execution).

Table 6: Simple Concurrent Automata Results

k	Search	KF Calls			Runtimes	
	Method	Max	Avg	Time	k-best	IMM
2	A*BC	17	5.1	0.98	0.05	
	A*	20	5.7	0.98	0.05	0.10
5	A*BC	35	18.8	0.98	0.17	
	A*	53	25.8	0.98	0.23	0.24
10	A*BC	85	39.6	0.97	0.38	
	A*	109	54.8	0.98	0.58	0.47
20	A*BC	150	77.7	0.97	0.76	
	A*	200	106.6	0.98	1.0	0.98

Note that the A*BC algorithm requires fewer Kalman filter calls to produce the same results as the A* driven estimator. Any additional overhead from tracking and learning the bounding conflicts is dominated by the time saved by reducing the number of Kalman filter executions needed as they are, by far, the most computationally expensive step of the estimator.

6.2 Ship Fluid System

This real-world example is the fluid portion of the shipboard fluid and electrical system described by Srivastava et al. (2008) and pictured in Figure 1. This system has sixty components with more than one operating mode (two tanks, four pumps, four non-vital loads, two vital loads, six check valves, thirty-two valves, and ten flow meters), resulting in over 10^{28} possible discrete modes for the entire system. This number of modes places the system well outside of standard techniques such as IMM estimators.

In this experiment, an A*BC based estimator was again compared to an A* based estimator on a variety of simulated runs of the fluid system. The focus on this experiment was on understanding the behavior of each method over time on a single timestep. The number of Kalman filter calls is used as a proxy for time in the following graphs as it is again, by far, the most expensive step in the current implementation, with approximately 93% of the total runtime spent deriving and executing the Kalman filter (with approximately 95% of that time spent in the symbolic solver to derive the filter on demand).

Figures 12 and 13 show two typical runs of the system. The first graph in each series shows the number of component distributions proven optimal for timestep $k + 1$ vs the number of Kalman filter executions. The second graph in each series shows the ratio between the likelihood of the first, most likely component distribution, and the latest produced component distribution vs the number of Kalman filter executions.

The third and fourth graphs in each series show the offset between the predicted and actual hybrid observation function values vs number of Kalman filter executions for A*BC and A*, respectively. These last two graphs show definitively that while both approaches have similarly high gaps between the bound's predicted cost and true cost at the beginning of the search, the A*BC approach learns from the high bounds and reduces the net error to be below that of standard A*. The error could be further reduced, by learning bounding conflicts involving multiple sensor readings. However, that approach comes at the cost of more time spent learning the bounding conflict and a higher branching factor when splitting on the conflict.

7. Conclusion

In this paper, we have introduced *bounding conflicts*, a novel extension of conflicts that describe both where a search algorithm's bounding function is not tight and a tighter bounding function for that region of the state space. Additionally, we have provided a best-first search and enumeration algorithm based on bounding conflicts (A*BC) and have described a state estimator for hybrid discrete/continuous systems built on top of this best-first enumerator.

The new state estimator based on A*BC outperforms the previous state of the art estimator for large-scale hybrid systems. It produces the best component estimates with fewer Kalman filter executions, allowing a better state estimate to be produced in less time.

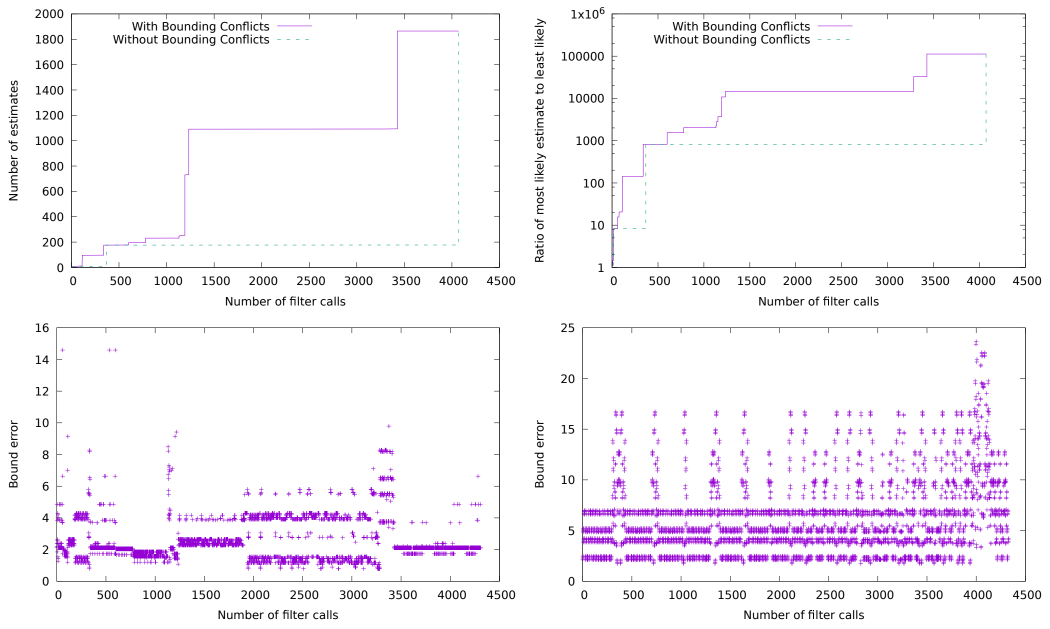


Figure 12: Set one of typical k-best filter performance on fluid system benchmark. Top left figure shows number of estimates vs filter calls. Top right figure shows ratio of most likely estimate to last estimate generated vs filter calls. Bottom left and right show offset between predicted and actual hybrid observation function values using A*BC and A*, respectively.

Future work will focus on composing bounding conflicts to get an even tighter bound. Additionally, tweaks to Weyl and more aggressive caching for filter derivations will be applied to reduce the amount of time spent in the symbolic solver to derive the filter on each Kalman filter execution.

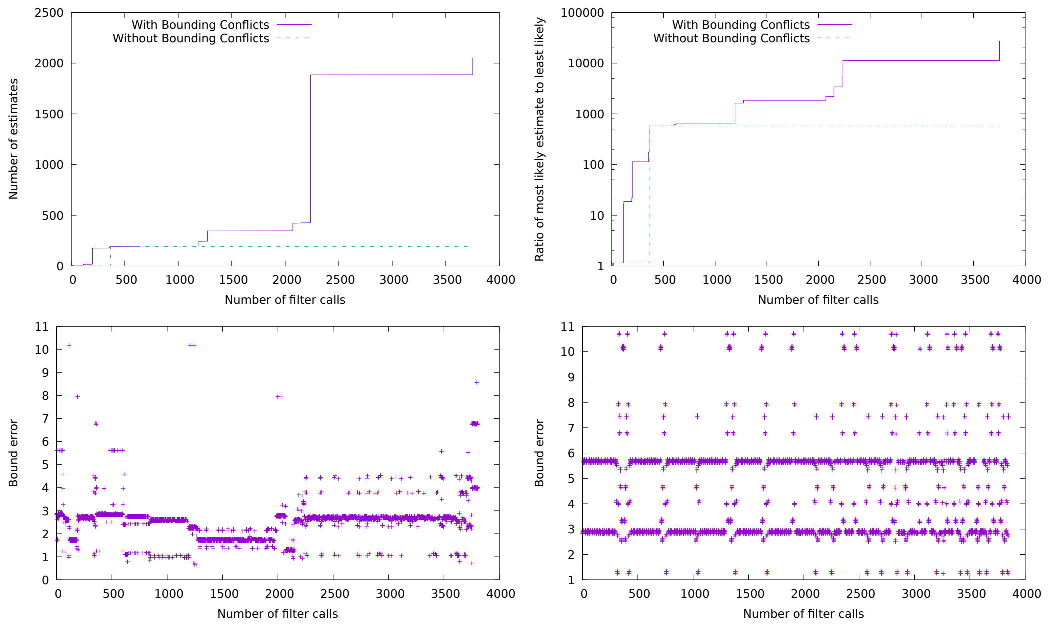


Figure 13: Set two of typical k-best filter performance on fluid system benchmark. Top left figure shows number of estimates vs filter calls. Top right figure shows ratio of most likely estimate to last estimate generated vs filter calls. Bottom left and right show offset between predicted and actual hybrid observation function values using A*BC and A*, respectively.

References

Ackerson, G., & Fu, K. (1970). On state estimation in switching environments. *IEEE Transactions on Automatic Control*, *15*(1), 10–17.

Blackmore, L., Funiak, S., & Williams, B. C. (2008). A combined stochastic and greedy hybrid estimation capability for concurrent hybrid models with autonomous mode transitions. *Journal of Robotic and Autonomous Systems*, *56*(2), 105–129.

Blom, H. A., & Bar-Shalom, Y. (1988). The interacting multiple model algorithm for systems with markovian switching coefficients. *IEEE transactions on Automatic Control*, *33*(8), 780–783.

Dago, P., & Verfaillie, G. (1996). Nogood recording for valued constraint satisfaction problems. In *Tools with Artificial Intelligence, 1996., Proceedings Eighth IEEE International Conference on*, pp. 132–139. IEEE.

De Kleer, J., & Williams, B. C. (1987). Diagnosing multiple faults. *Artificial intelligence*, *32*(1), 97–130.

Feldman, A., Provan, G., & Van Gemund, A. (2010). Approximate model-based diagnosis using greedy stochastic search. *Journal of Artificial Intelligence Research*, *38*, 371–413.

Garcia-Fernandez, Á. F., Morelande, M. R., & Grajal, J. (2012). Truncated unscented kalman filtering. *IEEE Transactions on Signal Processing*, *60*(7), 3372–3386.

Gehin, A.-L., Assas, M., & Staroswiecki, M. (2000). Structural analysis of system reconfigurability. *IFAC Proceedings Volumes*, *33*(11), 297–302.

Hanlon, P. D., & Maybeck, P. S. (2000). Multiple-model adaptive estimation using a residual correlation kalman filter bank. *IEEE Transactions on Aerospace and Electronic Systems*, *36*(2), 393–406.

Hofbaur, M. W., & Williams, B. C. (2002). Mode estimation of probabilistic hybrid systems. In *Hybrid Systems: Computation and Control*, pp. 253–266. Springer.

Hofbaur, M. W., & Williams, B. C. (2004). Hybrid estimation of complex systems. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, *34*(5), 2178–2191.

Julier, S. J., & Uhlmann, J. K. (1997). A new extension of the kalman filter to nonlinear systems. In *Int. symp. aerospace/defense sensing, simul. and controls*, Vol. 3, pp. 182–193. Orlando, FL.

Li, X. R. (2000). Multiple-model estimation with variable structure. ii. model-set adaptation. *IEEE Transactions on Automatic Control*, *45*(11), 2047–2060.

Li, X. R., Zwi, X., & Zwang, Y. (1999). Multiple-model estimation with variable structure. iii. model-group switching algorithm. *IEEE Transactions on Aerospace and Electronic Systems*, *35*(1), 225–241.

Li, X.-R., & Bar-Shalom, Y. (1996). Multiple-model estimation with variable structure. *IEEE Transactions on Automatic control*, *41*(4), 478–493.

Maybeck, P. S., & Stevens, R. D. (1991). Reconfigurable flight control via multiple model adaptive control methods. *IEEE Transactions on Aerospace and Electronic systems*, *27*(3), 470–480.

Muscettola, N., Nayak, P. P., Pell, B., & Williams, B. C. (1998). Remote agent: To boldly go where no ai system has gone before,. *Artificial Intelligence*, *103*(1-2), 5–48.

Narasimhan, S., & Brownston, L. (2007). Hyde–a general framework for stochastic and hybrid model-based diagnosis. *Proc. DX*, *7*, 162–169.

Nayak, P. (1995). *Automated Modeling of Physical Systems*. Lecture Notes in Artificial Intelligence. Springer.

Reinschke, K. J. (1988). Multivariable control: a graph theoretic approach..

Simon, D. (2010). Kalman filtering with state constraints: a survey of linear and nonlinear algorithms. *IET Control Theory & Applications*, *4*(8), 1303–1318.

Sorenson, H. W. (Ed.). (1985). *Kalman filtering: theory and application*. IEEE.

Srivastava, S., Cartes, D., Maturana, F., Ferrese, F., Pekala, M., Zink, M., Meeker, R., Carnahan, D., Staron, R., Scheidt, D., & Huang, K. (2008). A control system test bed for demonstration of distributed computational intelligence applied to reconfiguring heterogeneous systems. *Instrumentation Measurement Magazine, IEEE*, *11*(1), 30–37.

Teixeira, B. O., Tôrres, L. A., Aguirre, L. A., & Bernstein, D. S. (2010). On unscented kalman filtering with state interval constraints. *Journal of Process Control*, *20*(1), 45–57.

Thiébaux, S., Cordier, M.-O., Jehl, O., & Krivine, J.-P. (1996). Supply restoration in power distribution systems: A case study in integrating model-based diagnosis and repair planning. In *Proceedings of the Twelfth international conference on Uncertainty in artificial intelligence*, pp. 525–532. Morgan Kaufmann Publishers Inc.

Trave-Massuyes, L., & Pons, R. (1997). Causal ordering for multiple mode systems. In *Proceedings of the eleventh international workshop on qualitative reasoning*, pp. 203–214.

Tugnait, J. K. (1982). Detection and estimation for abruptly changing systems. *Automatica*, *18*(5), 607–615.

Williams, B. C., & Ragno, R. J. (2007). Conflict-directed A* and its role in model-based embedded systems. *Discrete Applied Mathematics*, *155*(12), 1562–1595.

Zippel, R. (1993). The weyl computer algebra substrate. *Design and Implementation of Symbolic Computation Systems*, 303–318.