Journal of Artificial Intelligence Research 67 (2020) 1-34 Submitted 11/2018; published 01/2020

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

**Eric M. 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

There is an increasing desire for autonomous systems to have high levels of robustness and safety, attained through continuously planning and self-repairing online. Underlying this is the need to accurately estimate the system state and diagnose subtle failures. Estimation methods based on hybrid discrete and continuous state models have emerged as a method of precisely computing these estimates. However, existing methods have difficulty scaling to systems with more than a handful of components. Discrete, consistency based state estimation capabilities can scale to this level by combining best-first enumeration and conflict-directed search. While best-first methods have been developed for hybrid estimation, conflict-directed methods have thus far been elusive as conflicts learn inconsistencies from constraint violation, but 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 enumeration algorithm based on bounding conflicts (A*BC) and a hybrid estimation method using this enumeration algorithm. Experiments show that an A*BC powered state estimator produces estimates up to an order of magnitude 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 by using planners online to generate novel responses to exceptional situations. A key capability needed by such systems is the ability to accurately estimate the system state. 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, these 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 these systems 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 and 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*.

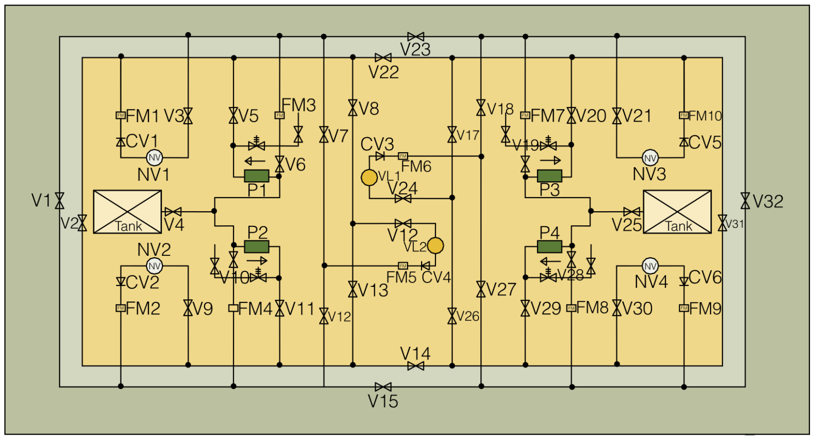
Purely discrete state estimators from the model-based reasoning community are able to scale in large part through the use of best-first enumeration and conflict-directed search (Williams & Ragno, 2007), as well as stochastic search methods (Feldman, Provan, & Van Gemund, 2010). Conflict-directed search 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 primary challenge is that a conflict is a consistency based concept that represents sets of states that have zero probability based on a proof of logical inconsistency. However, in the extreme case of a stochastic environment with unbounded uncertainty (such as Gaussian noise models), all behaviors are consistent, albeit unlikely.

One way to incorporate consistency-based conflicts into hybrid estimation is 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. However, this results in few conflicts being learned, so there is little to no improvement over not using conflicts.

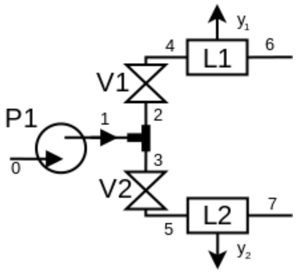
An ideal hybrid state estimation algorithm would be an any-time algorithm that quickly enumerates the most likely modes of the system and, given enough time, would produce all feasible modes. In this paper we present such an approach to hybrid estimation that augments best-first enumeration with a variant 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 used during search — is much lower than the true cost. Bounding conflicts are similar in spirit to *valued nogoods* (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 modes 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.

In the remainder of this paper we first summarize the problem of hybrid discrete and continuous state estimation and the concurrent probabilistic hybrid automata used to model hybrid systems. Then, we define a valued enumeration problem and describe hybrid state estimation as an instance of this problem. Next, we develop the A* with Bounding Conflicts (A*BC) search algorithm to solve valued enumeration problems. We then describe how bounding conflicts are learned from the hybrid state estimation problem. After walking through an example estimation problem, we last provide empirical data showing that A*BC enables a hybrid state estimator to provide estimates for real world problems that cover



(a) 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.



(b) Simple running example.

Figure 1: Motivating examples of hybrid systems.

significantly more probability mass in less time as compared to a state of the art, non-A*BC powered algorithm.

## 2. Hybrid State Estimation and Modeling

Hybrid state estimation is an instance of state filtering for systems with both discrete and continuous state. It takes as input a probabilistic model of the hybrid discrete and continuous system under observation, a prior probability distribution over the state of the system at time 0, and a sequence of observations and commands up to time $t$. From this, it computes the belief state of the system at time $t$. In this section, we introduce both the motivating problem behind this work and a pedagogical version of it that is used throughout the rest of the paper. Then, we review the hybrid state estimation problem and introduce the hybrid modeling scheme that this paper employs.

In this work, we focus on systems with a number of concurrently running processes. The dynamics of these processes are determined by their (discrete) operating mode and the processes interact through coupled continuous variables. We are interested in systems with a large number of these processes (therefore they have a large number of discrete mode variables) as that is where existing state of the art methods begin to break down. As a motivating example, consider a naval ship's electrical and cooling system, described by Srivastava, et al. (2008). The system, pictured in Figure 1a, 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 exchangers. Additionally, the loads report their current temperature.

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 would be made by a control program that relies on a state estimator to determine the likely configuration of the system.

This example system contains eighty-eight components modeled as concurrent processes (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) are trivial and have only one operating mode. The remaining sixty components have an average of three operating modes (for example, the valves can be open, closed, stuck open, and stuck closed), meaning the system has approximately $3^{60}$ (over $10^{28}$) possible discrete configurations it can be in.

As a pedagogical example, we present a simplified version of this cooling system, pictured in Figure 1b. 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. Additionally, the loads report their current temperature ($y_1$ and $y_2$).

## 2.1 Discrete Time Hybrid State Estimation

In this section, we formulate the discrete time hybrid state estimation problem as an instance of recursive Bayesian filtering.

The goal of a hybrid state estimator is to compute a probability distribution over the state of the system, otherwise known as the *belief state*, at time $t$ ($\mathbf{x}_t$) given a prior belief state ($\mathbf{x}_0$), observations of the system up to and including time $t$ ($\mathbf{y}_{1:t}$), and commands to the system up to and including time $t$ ($\mathbf{u}_{1:t}$). The state of the system is fully determined by the discrete mode variables ($\mathbf{m}_t$) and continuous state variables ($\mathbf{x}_{c,t}$), resulting in Equation 1 which describes the belief state.

$$p(\mathbf{m}_t, \mathbf{x}_{c,t} | \mathbf{y}_{1:t}, \mathbf{u}_{1:t}) \tag{1}$$

Due to the hybrid discrete and continuous nature of the state, a natural way to represent the belief state is as a mixture of $N$ independent sub-beliefs[1]. The $i$'th sub-belief at time $t$ ($\hat{\mathbf{x}}_t^{(i)}$) is parameterized by a weight ($w_t^{(i)}$), mode assignment for the entire system ($\hat{\mathbf{m}}_t^{(i)}$), and continuous probability distribution over every continuous state variable in the system ($p_{c,t}^{(i)}$). Intuitively, a sub-belief states that with probability $w_t^{(i)}$, the system is in the specified mode and the belief state over the continuous variables is described by the continuous distribution. The entire belief state can be constructed using a weighted sum over the sub-beliefs.

In the concurrent probabilistic hybrid automata modeling formalism (discussed in depth below), the control variables are, like the state variables, partitioned into discrete ($\mathbf{u}_d$) and continuous ($\mathbf{u}_c$) inputs, while the observations ($\mathbf{y}$) are restricted to only continuous values. The mode at any given time is dependent on the previous mode, previous continuous state, and discrete control inputs. The continuous state at any given time is dependent on the

1. In mixture models, these are normally called the mixture's component beliefs. We choose to use the term "sub-beliefs" to eliminate any confusion with the components that make up the system being estimated.

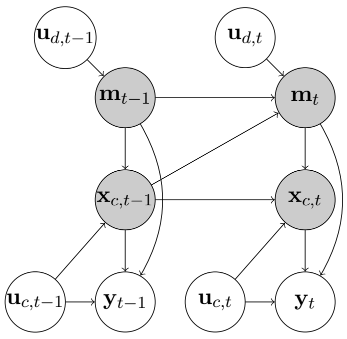


Figure 2: The recursive hybrid estimation problem shown as a dynamic Bayes net. The hidden variables are shaded.

previous continuous state, the current continuous control inputs, and the current mode of the system. The observations at any time are determined by the current continuous state and controls, as well as the current mode. The current mode affects the continuous state and observations by determining which equations are used to relate those values to their other dependencies. This set of partitions and dependencies gives rise to a set of conditional probability distributions visualized in the dynamic Bayes net in Figure 2.

Given these dependencies and partitions, Hofbaur and Williams (2004) have shown that a sub-belief $\hat{\mathbf{x}}_t^{(j)}$ can be recursively computed from a sub-belief at the previous time step $\hat{\mathbf{x}}_{t-1}^{(i)}$ and a new mode assignment $\hat{\mathbf{m}}_t^{(j)}$. The weight computation is shown in Equations 2 and 3, using the *hybrid transition function* defined in Equation 4 and the *hybrid observation function* defined in Equation 5 and where $\eta$ is a normalization factor.

$$w_{t|t-1}^{(j)} = P_T(\hat{\mathbf{m}}_t^{(j)}, \hat{\mathbf{x}}_{t-1}^{(i)}, \mathbf{u}_{d,t}) w_{t-1}^{(i)} \tag{2}$$

$$w_t^{(j)} = \eta P_O(\mathbf{y}_t, p_{c,t}^{(j)}, \mathbf{u}_{c,t}) w_{t|t-1}^{(j)} \tag{3}$$

$$P_T(\hat{\mathbf{m}}_t^{(j)}, \hat{\mathbf{x}}_{t-1}^{(i)}, \mathbf{u}_{d,t}) \triangleq \int_{\mathbf{x}_c} p_{c,t-1}^{(i)}(\mathbf{x}_c) p(\hat{\mathbf{m}}_t^{(j)} | \hat{\mathbf{x}}_{t-1}^{(i)}, \mathbf{u}_{d,t}) \tag{4}$$

$$P_O(\mathbf{y}_{c,t}, \hat{\mathbf{x}}_{t-1}^{(i)}, \mathbf{u}_{c,t}) \triangleq p(\mathbf{y}_{c,t} | \hat{\mathbf{m}}_t^{(j)}, \hat{\mathbf{x}}_{t-1}^{(i)}, \mathbf{u}_{c,t}) \tag{5}$$

## 2.2 Probabilistic Hybrid Automata

As previously discussed, in this work we focus on systems with a large number of components (and therefore a large number of discrete variables), representing a set of concurrent processes. As such, we use concurrent probabilistic hybrid automata (CPHA) (Hofbaur & Williams, 2004) to model systems for the hybrid state estimator. 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.

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

**Definition 1.** *A probabilistic hybrid automaton (PHA) is described by the 9-tuple* $\langle \mathbf{x}, \mathcal{X}_d, \mathbf{w}, U_d, F, \mathcal{F}_{DE}, \mathcal{F}_{AE}, T, \mathcal{T} \rangle$*, where:*

- $\mathbf{x} \triangleq \{x_d\} \cup \mathbf{x}_c$ is the automaton's state variables. $x_d$ denotes the discrete operating mode and has a domain of $\mathcal{X}_d$. $\mathbf{x}_c$ denotes $n_x$ continuous state variables.
- $\mathbf{w} \triangleq \{u_d\} \cup \mathbf{w}_c \cup \mathbf{v}_c$ denotes the *interface* variables of the automaton. $u_d$ is the discrete valued command (input only) variable with domain $U_d$. $\mathbf{w}_c$ is the set of $n_w$ continuous interface (input or output) variables. $\mathbf{v}_c$ is the set of $n_v$ Gaussian noise variables (input only).
- $F : \mathcal{X}_d \to \mathcal{P}(\mathcal{F}_{DE} \cup \mathcal{F}_{AE})$ specifies the continuous evolution of the automaton over discrete time by mapping an operating mode to a finite set of first order, discrete time difference equations drawn from $\mathcal{F}_{DE}$ and algebraic equations drawn from $\mathcal{F}_{AE}$. Both sets of equations are defined over $\mathbf{x}_{c,t}$ (the values of the continuous state variables at time $t$), $\mathbf{x}_{c,t-1}$ (the values of the continuous state variables at time $t-1$), $\mathbf{w}_{c,t}$ (the values of the continuous I/O values at time $t$), and $\mathbf{v}_{c,t}$ (the values of the continuous noise inputs at time $t$).
- $T : \mathcal{X}_d \to \mathcal{P}(\mathcal{T})$ specifies the evolution of the automaton's operating mode over discrete time. It maps an operating mode to a finite set of guarded transitions drawn from $\mathcal{T}$. Each guarded transition in $\mathcal{T}$ consists of a set of probabilities of transitioning to each mode in $\mathcal{X}_d$ and a guard defined over $\mathbf{x}_{c,t-1}$, $\mathbf{u}_{d,t}$, and $\mathbf{w}_{c,t}$. The transition is enabled if and only if the guard is satisfied. For any mode, the guards of the selected transitions must completely partition $U_d \times \mathbb{R}^{n_w} \times \mathbb{R}^{n_x}$

### 2.2.1 Example PHA - Electrical Load

We now describe an electrical load with heat exchanger ($L_1$) from the pedagogical example fluid system as a PHA. It has one mode variable ($m_{L_1}$) that can take the values "on" and "off". Its continuous state is defined by its temperature ($T_1$). It has one discrete command input ($u_{L_1}$) that can take the values "on" and "off". It interfaces with the other components ($\mathbf{w}_c$) through the pressure and flow of water at both ends, as well as with its temperature sensor ($p_4$, $p_6$, $f_4$, $f_6$, and $y_1$). It interfaces with the environment ($\nu_c$) through two noise inputs: the sensor noise $\nu_{o_1}$ and process noise $\nu_{p_1}$.

Table 1 summarizes the $F$ function. In both modes, there is conservation of the flow, the sensor output measures the temperature corrupted by additive Gaussian noise, and there is a resistance of $R_l$ to the flow. The discrete time difference equations show how the temperature changes in relation to the temperature of the water ($T_W$, a constant), the efficiency and heat capacities (bundled together in $k_1$ and $k_2$, both constants), and the rate of flow. Additionally, when on, the load produces extra heat ($Q$).

Last, Figure 3 shows $T$ for the automaton in graphical form. The condition $c_1$ is true if $u_{L_1} = \text{off}$, $c_2$ is true if $u_{L_1} = \text{on}$, and $c_3$ is true if $T_1 > 350K$. The transition func-

Table 1: **F** of load. Note that both modes share the first three equations pertaining to conservation of flow, pressure drop, and output, but differ in the final equation describing the temperature change over time.

| $\mathbf{x}_d$ | **F** |
|---|---|
| on | $f_4 + f_6 = 0$<br>$p_4 - p_6 = R_l f_4$<br>$y_{1,t} = T_{1,t} + \nu_{o_1,t}$<br>$T_{1,t} = T_{1,t-1} + k_2^{-1}(-k_1(T_{1,t-1} - T_W)f_4 + Q)\Delta t + \nu_{p_1,t}$ |
| off | $f_4 + f_6 = 0$<br>$p_4 - p_6 = R_l f_4$<br>$y_{1,t} = T_{1,t} + \nu_{o_1,t}$<br>$T_{1,t} = T_{1,t-1} + k_2^{-1}(-k_1(T_{1,t-1} - T_W)f_4)\Delta t + \nu_{p_1,t}$ |

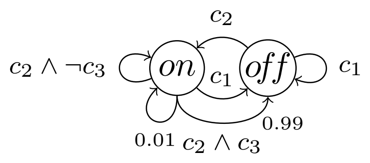


Figure 3: Load transition diagram.

tion demonstrates the load following commands, but, as a fail-safe, it will autonomously transition off if the temperature gets too high.

## 2.3 Concurrent Probabilistic Hybrid Automata

At the system level, the artifact being monitored is a composition of PHAs operating concurrently. While a composition of PHAs 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. 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.

- $\mathbf{u} \triangleq \{\mathbf{u}_d, \mathbf{u}_c\}$ 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 every discrete-valued command variable. $\mathbf{u}_c$ is a subset of the union of every continuous interface ($\mathbf{w}_c$) set.

- $\mathbf{y}_c$ 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 ($\mathbf{w}_c$) sets.

Given an assignment for every mode variable, the system wide continuous dynamics and outputs can be computed, typically with a symbolic solver, as equations of the form below. Equation 6 represents the evolution of the system-wide continuous state and Equation 7 represents the continuous observations of the system.

$$\mathbf{x}_{c,t} = \mathbf{f}_t(\mathbf{x}_{c,t-1}, \mathbf{u}_{c,t}, \mathbf{v}_{s,t}) \tag{6}$$

$$\mathbf{y}_{c,t} = \mathbf{g}_t(\mathbf{x}_{c,t}, \mathbf{u}_{c,t}, \mathbf{v}_{o,t}) \tag{7}$$

In the pedagogical 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 conserved (flow is defined to be positive going away from the pump). 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 temperature outputs from the two loads.

In the example system, when the pump is on, the valves are open, and the loads are on, the set of solved equations is provided by Equations 8 and 9.

$$\begin{pmatrix} T_{1,t} \\ \\ T_{2,t} \end{pmatrix} = \begin{pmatrix} \left( \dfrac{-k_1(T_{1,t-1} - T_W)P}{k_2(R_v + R_l)} + \dfrac{Q}{k_2} \right) \Delta t \\ \left( \dfrac{-k_1(T_{2,t-1} - T_W)P}{k_2(R_v + R_l)} + \dfrac{Q}{k_2} \right) \Delta t \end{pmatrix} + \begin{pmatrix} \nu_{p_1,t} \\ \\ \nu_{p_2,t} \end{pmatrix} \tag{8}$$

$$\begin{pmatrix} y_{1,t} \\ y_{2,t} \end{pmatrix} = \begin{pmatrix} T_{1,t} \\ T_{2,t} \end{pmatrix} + \begin{pmatrix} \nu_{o_1,t} \\ \nu_{o_2,t} \end{pmatrix} \tag{9}$$

### 2.4 Estimation of Continuous State for Concurrent Probabilistic Hybrid Automata

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 given PHA's equations (e.g., a valve in the closed state completely determines the flow rates on both ends), but in another mode it may be a required input to the same PHA's set of equations (e.g., a valve in the open state requires other components to determine the flow rate). However, given a mode assignment to every component, 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.[2] This analysis is a critical component of bounding conflict extraction, discussed further in section 5.

Once the system-wide dynamics and observation equations are determined, any suitable continuous state estimator can be used to compute both the continuous state distribution and the observation likelihood for a sub-belief. Commonly used continuous filters for this problem are Kalman filters, Kalman filter variants such as the Extended Kalman

2. 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.

Filter (EKF) (Hofbaur & Williams, 2004; Sorenson, 1985) and Unscented Kalman Filter (UKF) (Julier & Uhlmann, 1997), and particle filters (Blackmore et al., 2008; Doucet, 1998).

The experimental section of this paper uses a Kalman filter variant as its continuous estimator. If such a filter is used, the observation likelihood ($P_O$) is determined using the residual of the expected measurements ($\widetilde{\mathbf{y}}_t^{(j)}$) and the computed covariance of this residual ($\mathbf{S}_t^{(j)}$). The probability of this observation[3] is shown in Equation 10.

$$P_O(\mathbf{y}_{c,t}, \hat{\mathbf{x}}_{t-1}^{(j)}, \mathbf{u}_{c,t}) = \frac{1}{|2\pi \mathbf{S}_t^{(j)}|^{1/2}} e^{-0.5(\widetilde{\mathbf{y}}_t^{(j)})^\top (\mathbf{S}_t^{(j)})^{-1} \widetilde{\mathbf{y}}_t^{(j)}} \tag{10}$$

While this is the strictly correct form of $P_O$ for Kalman filters, Maybeck and Stevens (1991) suggest dropping the normalization constant in multiple model methods 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 11.

$$P_O(\mathbf{y}_{c,t}, \hat{\mathbf{x}}_{t-1}^{(j)}, \mathbf{u}_{c,t}) = e^{-0.5(\widetilde{\mathbf{y}}_t^{(j)})^\top (\mathbf{S}_t^{(j)})^{-1} \widetilde{\mathbf{y}}_t^{(j)}} \tag{11}$$

## 3. Best-First Enumeration and Hybrid State Estimation

The previous section defined the hybrid state estimation problem and described a modeling formalism used for systems with a large number of concurrently operating components. This section defines a best-first enumeration problem and describes how such an enumerator is used in a $k$-best hybrid state estimator. The following section will then develop the A*BC algorithm which acts as such an enumerator.

As previously discussed, this work represents a belief state as a mixture of sub-beliefs. Additionally, Equations 2 and 5 show how a sub-belief at time $t$ can be computed from a sub-belief at time $t-1$, given a mode for time $t$. Given that computing every single sub-belief at time $t$ is intractable, the as yet unanswered question is how to choose which sub-beliefs to extend and which mode assignments to use when extending them. Two general methods for doing this are stochastic sampling and greedy enumeration. This paper focuses on $k$-best enumeration methods, a subclass of greedy enumeration.[4]

Hybrid estimators that use $k$-best enumeration (otherwise known as $k$-best hybrid state estimators) approximate the belief at a given time using only the $k$ sub-beliefs that have the highest weights and assume that other mode assignments have weights of zero. $k$ may be a constant or dynamically determined at each time step (e.g., a fixed amount of time is used to enumerate sub-beliefs or sub-beliefs are produced until a minimum probability mass threshold is met). A $k$-best hybrid state estimator will produce sub-beliefs at time $t$ by framing a valued enumeration problem, defined below, and then using a best-first enumerator to enumerate the $\hat{\mathbf{x}}_{t-1}^{(i)}$ and $\hat{\mathbf{m}}_t^{(j)}$ pairs that result in the highest values of $w_t^{(j)}$.

3. 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 3 occurs.

4. 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.

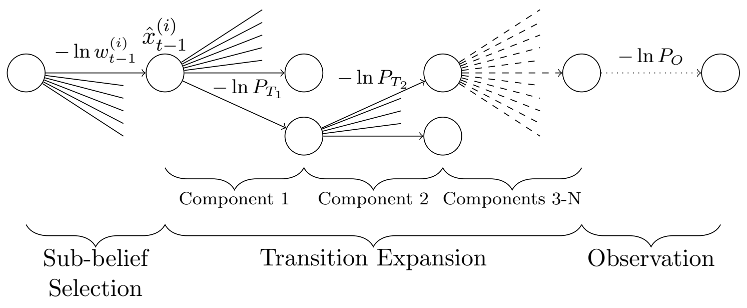


Figure 4: Search tree for hybrid state estimation.

**Definition 3.** *A valued enumeration problem is described by the tuple* $\langle Y, D, c\rangle$ *, 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.

*The solution to a valued enumeration problem is a prefix of the complete list of full assignments to* $Y$*, when ordered by increasing* $c$*.*

Formally, a $k$-best hybrid state estimator defines a valued enumeration problem where the domain of the first variable is the set of sub-beliefs from the previous time step: $d_1 = \bigcup_i \hat{\mathbf{x}}_{t-1}^{(i)}$. The domains of the remaining variables are the mode domains for every component in the CPHA: $\{d_2, \ldots, d_m\} = \{\mathcal{X}_{d,1}, \ldots, \mathcal{X}_{d,N}\}$. The cost function takes these assignments, computes the new continuous distribution and weight, then returns $-\ln w_t^{(j)}$. The logarithm is used to avoid numerical underflow when dealing with low probabilities and it is negated to fit into a cost based framework. This decision problem is framed at every time step $t$, the enumerator is run until a prefix of length $k$ is produced (where $k$ is either fixed or dynamically determined as described above), and the enumerated sub-beliefs are used for $d_1$ in the next enumeration problem at time $t + 1$.

Assuming a search-based enumerator (such as the one described in the following section), the search performed by the enumerator is visualized in Figure 4. The first step in the search is choosing a sub-belief from the previous time step to extend. The cost of each choice is the negative log of its weight. The next steps of the search then assign modes to every component to determine the full mode of the system at the current time step. Assuming that components transition independently, the cost of each of these steps is the negative log of the selected transition, given the continuous belief and starting mode implied by the selection of $\hat{\mathbf{x}}_{t-1}^{(i)}$. The last step of the search is a *pseudo* step in the pictured search tree. It is there to visualize the accumulation of cost from the observation likelihood after all modes are assigned, but does not correspond directly to any decision variables.

The total cost used by the enumerator is computed by summing all individual costs, resulting in the following cost function. This function is the negative log of Equations 2 and 3, under the assumption of independent mode transitions.

$$c(Y) = -\ln w_t^{(j)} = -\ln w_{t-1}^{(i)} - \sum_c \ln P_{T_c}(\hat{m}_{c,t}^{(j)}, \hat{\mathbf{x}}_{t-1}^{(i)}, \mathbf{u}_t) - \ln P_O(\mathbf{y}_{c,t}, \hat{\mathbf{x}}_{t-1}^{(i)}, \mathbf{u}_{c,t}) \qquad (12)$$

## 4. Best-First Enumeration Using Bounding Conflicts

The previous section defined the valued enumeration problem and demonstrated 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. Additionally, we describe bounding conflicts themselves and provide a method for learning bounding conflicts in the hybrid estimation domain.

The A* with Bounding Conflicts algorithm builds upon the Constraint-based A* and Conflict-directed A* algorithms (Williams & Ragno, 2007). Given a valued enumeration problem, A*BC searches through the space of partial assignments to the variables. Each member of the search space has every partial and full assignment that is a superset of the current partial assignment as its neighbors.

**Definition 4.** *An A*BC search problem is described by the three tuple* $\langle VE, f, L\rangle$ *, where :*

- $VE$ is a valued enumeration problem.
- $f : \mathcal{P}(Y) \to \mathbb{R}$ is the bounding function. $f$ maps partial assignments to the decision variables to a lower bound on the cost of any full assignment that is a superset of the input.
- $L : \mathcal{P}(Y) \to \Gamma$ is a function that returns a (potentially empty) set of bounding conflicts learned from a partial assignment.

The bounding function is typically the same form as A*'s bounding function: $f(y) = g(y) + h(y)$, where $g$ represents the cost accumulated by the assignments so far and $h$ is a lower bound on the cost that will be incurred from assigning the remaining variables. For the $k$-best hybrid estimation problem, at some search node where the modes for components 1 through $C$ have been determined, the $g$ part of the bounding function is the cost accrued by choosing the sub-belief from time $t-1$ to expand and the new modes for components 1 through $C$. The cost to go is determined by assuming each component transitions into its most likely mode and that the observations perfectly match the expected observations.

$$f(Y) = g(Y) + h(Y) \tag{13}$$

$$g(Y) = -\ln w_{t-1}^{(i)} - \sum_{c=1}^{C} \ln P_{T_c}(\hat{m}_{c,t}^{(j)}, \hat{\mathbf{x}}_{t-1}^{(i)}, \mathbf{u}_t) \tag{14}$$

$$h(Y) = -\sum_{c=C+1}^{N} \ln \max_{\hat{m}_{c,t}^{(j)}} P_{T_c}(\hat{m}_{c,t}^{(j)}, \hat{\mathbf{x}}_{t-1}^{(i)}, \mathbf{u}_t) - \ln 1 \tag{15}$$

If the most likely transition probability for a component is difficult to compute, assuming a value of 1 for $P_{T_c}$ is admissible as well. However, in many systems this transition is easy to compute exactly for every component. Therefore, the gap between the bounding function and actual best extension to the partial assignment is due to the assumption of observations perfectly matching the expected values.

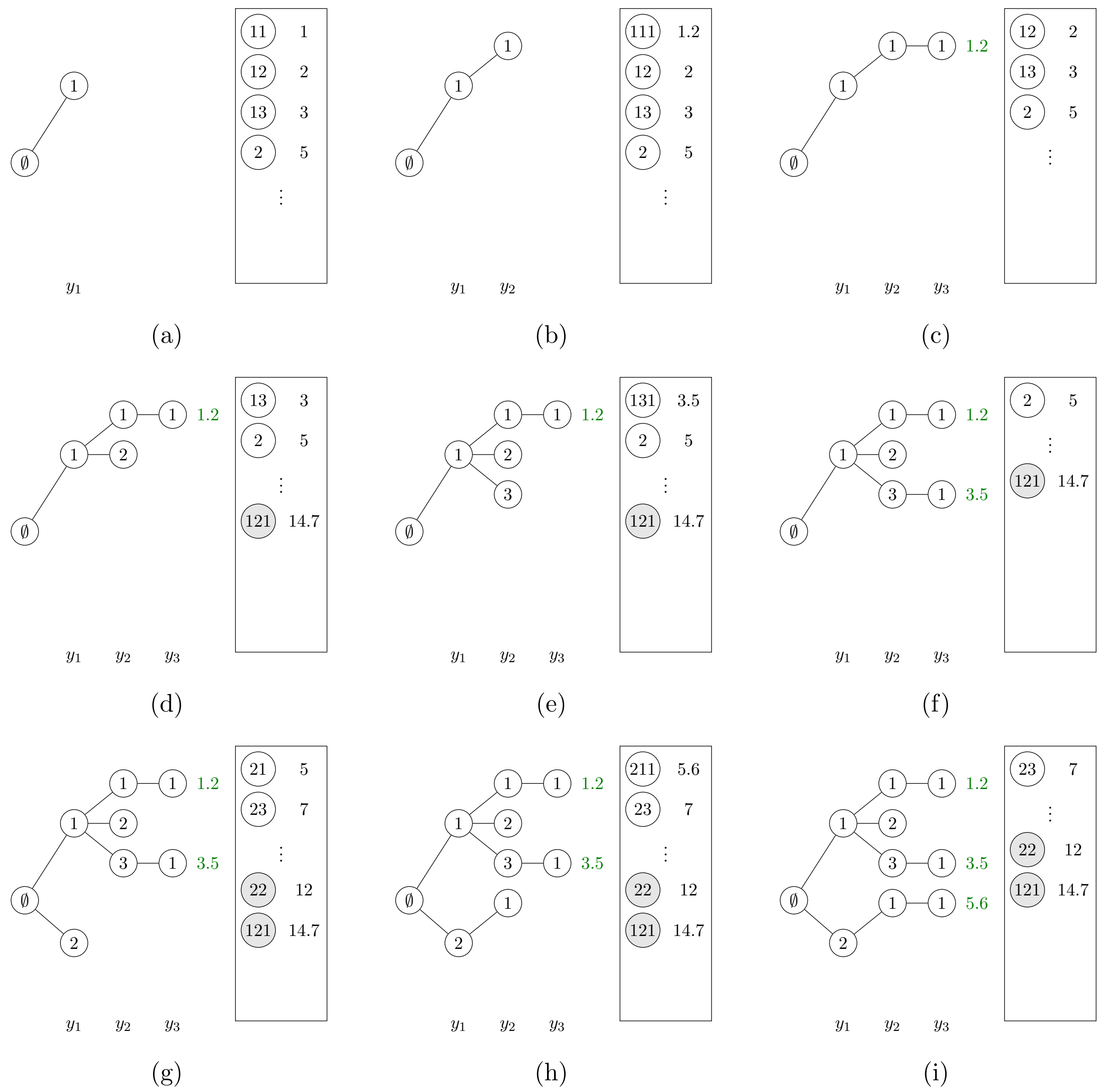


Figure 5: A*BC in a nutshell, applied to a problem with three variables, $y_1$, $y_2$, and $y_3$ with domains $\{1, 2\}$, $\{1, 2, 3\}$, and $\{1\}$.

### 4.1 A*BC in a Nutshell

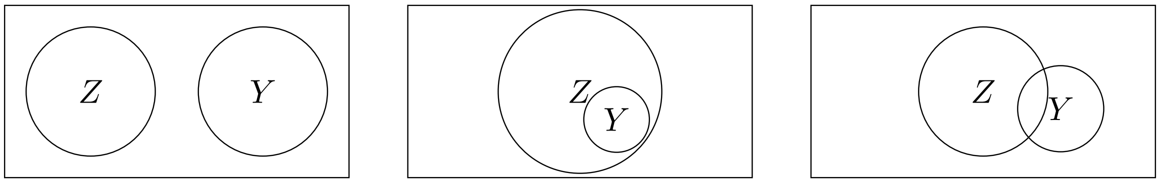
A*BC is an optimal, best-first enumeration algorithm. It is designed for problems where a weak bounding function is known at the beginning of the search, but there are opportunities during the search to learn tighter bounding functions that apply to some subset of the states. A*BC incorporates these tighter bounding functions as they become known by updating the bounding function and changing the order in which neighboring states are explored.

In the case of hybrid state estimation, the weakness of the bounding function is due to assuming the best case scenario that the actual observations perfectly match the expected observations. For any given partial assignment, it is exceedingly rare for any of its extensions to have a $P_O$ value of 1, resulting in the bounding function routinely underestimating the true cost by a large amount. However, it is hard to get a more accurate bound because computing the true $P_O$ involves running a continuous estimator that could non-trivially depend on the new modes for every component in the CPHA. Instead, a better bounding function is learned during the search by extracting extra information during the computation of $P_O$.

The information about a tighter bounding function gathered during the search is represented using *bounding conflicts*. A bounding conflict represents both a tighter bounding function and a compact encoding of the regions of the state space where it is valid. 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 full assignments covered by a bounding conflict, one child of the node is generated to represent this 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 procedure is illustrated at a high level by Figure 5. This toy problem has three variables (for simplicity, the domain of variable $y_3$ has only one element). The current search queue (along with the $f$ value for each node in the queue) is visualized in the box. Once a node is popped from the queue it is placed in the search tree and has its children placed on the queue. Every full assignment has the cost of that assignment placed next to it in the search tree. In (c), the best child of node 12 is believed to have a cost of 2, however once all the children are expanded the best child is found to be 121 with a cost of 14.7 (shown in (d)). At this point, a conflict learning function is used to learn why the earlier estimate was off and try to generalize it. Suppose that the conflict learning function determines that $f$ underestimates the cost of any partial assignment containing $y_2 = 2$ by 6. This allows A*BC to better estimate the cost of any node with $y_2 = 2$ (shown as shaded nodes) to avoid expanding them until necessary.



(a) Resolves the bounding conflict. (b) Manifests the bounding conflict. (c) Partially manifests and partially resolves.

Figure 6: The three potential relationships between a partial assignment $y$ and a bounding conflict with partial assignment $z$. Each circle represents the set of full assignments to which the corresponding partial assignments could be extended.

## 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 5.** *A bounding conflict is a pair* $\langle z, b \rangle$ *where:*

- $z$ is a partial assignment to the decision variables of a best-first enumeration problem and
- $b : P_z \rightarrow \mathbb{R}$ is a function that maps extensions of $z$ to a bound on the cost of any further extensions.

Given a search node with partial assignment $y$ and a bounding conflict with partial assignment $z$, there are three ways they can be related, as illustrated in Figure 6. In the following, $Y$ refers to the set of full extensions to partial assignment $y$ and $Z$ refers to the set of full extensions to partial assignments of $z$. If $Y$ does not intersect with $Z$, the search node is said to *resolve* the bounding conflict (Figure 6a). If $Y$ is a subset of $Z$, the search node is said to *manifest* the bounding conflict (Figure 6b). Otherwise if $Y$ intersects with $Z$ but is not a subset, the search node *partially manifests and partially resolves* the bounding conflict.

The key takeaway is that if a partial assignment manifests a bounding conflict, then $b$ represents a valid bounding function for that partial assignment (i.e., it underestimates the cost of any full extension), in addition to the bounding function known at the beginning of the search ($f$). This allows the A*BC algorithm to evaluate multiple valid bounds to determine the tightest one to use when sorting its search queue.

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. As infeasibility can be represented as an infinitely high cost, the tighter bound for a classical conflict is $b(x) = \infty$.

## 4.3 A*BC Search Algorithm

In this subsection, we describe the A*BC search algorithm in depth, but first we describe the notation used. A*BC is a search algorithm and every search node, $sn$, it considers is described by a partial assignment $s$ and a set of manifested bounding conflicts $\beta$. Dot

$$\begin{tikzcd}
\{A=G, B=G\} & & \{A=U, B=G\} \\
& \{B=G\} \arrow[ul] \arrow[ur] & \\
\{A=G\} \arrow[uu] \arrow[dd] & \{\} \arrow[l] \arrow[r] \arrow[u] \arrow[d] \arrow[uul] \arrow[uur] \arrow[ddl] \arrow[ddr] & \{A=U\} \arrow[uu] \arrow[dd] \\
& \{B=U\} \arrow[dl] \arrow[dr] & \\
\{A=G, B=U\} & & \{A=U, B=U\}
\end{tikzcd}$$

Figure 7: The complete search space for a best-first enumeration problem with two variables (A and B) that can each take on the value (G)ood or (U)nknown.

notation is used to refer to these properties of a given search node, so $sn.\beta$ refers to the manifested bounding conflicts of node $sn$. Individual bounding conflicts are denoted using $\gamma$. Again, dot notation is used, so $\gamma.z$ refers to the partial assignment of bounding conflict $\gamma$, and $\gamma.b$ refers to the bounding conflict's bounding function. Last, A*BC learns bounding conflicts while searching and the set of known bounding conflicts is represented by $\Gamma$.

The A*BC algorithm performs best-first enumeration by framing it as a graph search problem. The nodes on the graph are every partial and complete assignment to the decision variables. The starting node is the empty assignment and the goal nodes are every full assignment. The neighbors of each node are every extension to that node's partial assignment. This is the same graph search framing used by Constraint-Based A* and Conflict-Directed A*. This graph is visualized for a two variable/two value problem in Figure 7.

In this definition of the search space, there are many different paths leading to each solution. For instance, if there are $m$ variables and we consider all paths that assign just one variable at a time, there are $m!$ distinct, equal cost paths leading to every goal node. The number of paths only grow as assigning multiple variables simultaneously is considered. To deal with this extremely large number of valid paths, A*BC ensures that it considers only one path to every goal node. It accomplishes this by visiting only a subset of the neighbors for each search node. The sets of full assignments reachable from the visited neighbors are guaranteed to be exhaustive and mutually exclusive. This principle of following only a single path to each full assignment has also been previously used by Constraint-Based A* and Conflict-Directed A*.

```
Input: The node to expand, $sn$.
Output: The mutually exclusive children of $sn$ that assign one additional variable.
1 children ← {};
2 $x$ ← choose unassigned variable in $sn.s$;
3 foreach $v \in dom(x)$ do
4     children ← children ∪ {Make-Node($s = sn.s \cup \{x = v\}, \beta = sn.\beta$)};
5 end
6 return children;
```

**Algorithm 1:** Neighbors-By-Variable

Given a node in the graph corresponding to a partial assignment $y$, A*BC chooses the neighbors in one of two ways. One method, called Neighbors-By-Variable first chooses a variable and then returns all neighbors that extend $y$ by adding an assignment for the chosen variable. This algorithm is shown in Algorithm 1. The variable to assign can be chosen (line 2) using any method appropriate to the problem at hand, such as ordering using a lexicographic order, ordering based on domain size, choosing based on how often variables are involved in bounding conflicts, etc. This method of generating neighbors is used when a partial assignment either resolves or manifests all known bounding conflicts.

```
   Input: The node to expand, $sn$.
   Input: The bounding conflict to use, $\gamma$.
   Output: Mutually exclusive neighbors of $sn$, one that manifests $\gamma$ the others resolve $\gamma$.
 1 children $\leftarrow$ {Make-Node($s = sn.s \cup \gamma.z, \beta = sn.\beta \cup \{\gamma\}$)};
 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 $\leftarrow$ Make-Node($s = sn.s \cup \{x = v\} \cup \phi, \beta = sn.\beta$) $\cup$ children;
 8     end
 9     $\phi \leftarrow \phi \cup \{x = y\}$;
10 end
11 return children;
```

**Algorithm 2:** Neighbors-By-BC

If there is a known bounding conflict that a search node partially manifests and partially resolves, then a different method is used to choose the neighbors. This algorithm, Neighbors-By-BC, is shown in Algorithm 2. Given a search node and a bounding conflict, this method chooses the neighbors such that every neighbor either completely resolves or completely manifests the bounding conflict, with special care taken such that the set of full assignments represented by each neighbor remain disjoint.

The first neighbor generated by this method is the extension of the search node's partial assignment to include the partial assignment contained in the bounding conflict (line 1). This first neighbor is the only neighbor that manifests the bounding conflict.

To generate the remaining neighbors, Neighbors-By-BC iterates over the assignments $(x = y)$ in the bounding conflict and resolves the conflict by making search nodes that contain every assignment to $x$ except $y$ (lines 3-8). This construction ensures that every neighbor will resolve the bounding conflict and remain disjoint from the first neighbor. However, the neighbors need to remain disjoint from each other as well. To ensure this, once every value in $(dom(x) - \{y\})$ is exhausted and another assignment from the bounding conflict is chosen, the assignment $x = y$ is added to every remaining neighbor (lines 9 and 7). This keeps all neighbors disjoint from each other by construction.

While a weak bounding function $f$ is known at the start of the algorithm, the bounding conflicts learned during the enumeration allow A*BC to compute tighter bounds over time. The process used to compute bounds (A*BC-Bound) is shown in Algorithm 3. The tightest bound is determined by taking the maximum value produced by the original bounding function and the bounding functions of all manifested bounding conflicts.

```
Input: A search node, $sn$.
Input: The original, weak bounding function $f$.
Output: The tightest known bound on cost for extensions to the node's partial
        assignment.
1 bound $\leftarrow f(sn.s)$;
2 foreach $\gamma \in sn.\beta$ do
3     bound $\leftarrow \max(\text{bound}, \gamma.b(sn.s))$;
4 end
5 return bound;
```

**Algorithm 3:** A*BC-Bound

The methods of generating neighbors and computing tighter bounds from bounding conflicts are brought together in a single A*-like algorithm, as described by Algorithm 4. A*BC maintains an open list of search nodes representing its current queue. This queue is initialized with a search node representing the empty partial assignment (line 1). At every iteration, it removes the search node with the current best bound from the queue (lines 5-6). If it represents a full assignment, it is known to be optimal and is placed on the list of solutions to be returned when the algorithm is terminated (line 8).

If the node contains only a partial solution, then the set of bounding conflicts is searched to find the bounding conflicts that the node either manifests ($\Gamma_m$) or partially manifests ($\Gamma_p$) (line 10). If the node manifests more bounding conflicts than it did the last time it was seen, the set of manifested bounding conflicts is updated, the search node is requeued, and the iteration repeats (lines 11-15). This requeueing allows the bound for the node to be updated based on the latest available information (bounding conflicts). If the search node is still the best, it will be immediately reconsidered and expanded, otherwise it is pushed further down the search queue, preventing the queue from growing from the addition of unpromising neighbors.

If the manifested bounding conflicts of the search node do not need to be updated, its neighbors are generated. Again, Neighbors-By-BC is used when the node partially manifests some bounding conflicts (lines 16-18) and Neighbors-By-Variable is used otherwise (lines 19-20).

Once the neighbors are generated, they are added to the open queue (line 23). Additionally, each neighbor is analyzed by the user provided bounding conflict learning function to determine if calculating the $f$ value for that node revealed any information that can be used to improve the bounds of other similar nodes. If so, the returned bounding conflicts are incorporated into the global set of known bounding conflicts (line 24).

Once the algorithm's termination condition is met (such as a timeout or the required number of solutions have been computed), the enumerated solutions are returned.

## 5. Learning Bounding Conflicts for Hybrid Estimation

As discussed previously, a bounding conflict describes a tighter bounding function $b$ and a region where the tighter bounding function is valid in the form of a partial assignment $z$. Additionally, the original bounding function used in $k$-best hybrid estimation is weak because it assumes that the observations perfectly match the expected observations. This

```
Input: A Best-First Enumeration Problem.
Input: A bounding function $f$.
Input: A bounding conflict learning function $L$.
Output: The set of solutions proven optimal before termination.
open ← {Make-Node(s={}, $\beta$={})};
solutions ← {};
$\Gamma$ ← {};
while not Terminate?(solutions, open) do
    $sn \leftarrow \text{argmin}_{sn_i \in \text{open}}$ A*BC-Bound($sn_i$);
    open ← open − {$sn$};
    if Full-Assignment?($sn$) then
        solutions ← solutions ∪ {$sn$};
    else
        $\Gamma_p, \Gamma_m$ ← Find-Relevant-Conflicts($sn, \Gamma$);
        if $\Gamma_m - sn.\beta$ not empty then
            $sn.\beta \leftarrow sn.\beta \cup \Gamma_m$;
            open ← open ∪ {neighbor};
            continue;
        end
        if $\Gamma_p$ not empty then
            $\gamma$ ← an element of $\Gamma_p$;
            neighbors ← Neighbors-By-BC($sn, \gamma$);
        else
            neighbors ← Neighbors-By-Variable($sn$);
        end
        foreach neighbor ∈ neighbors do
            open ← open ∪ {neighbor};
            $\Gamma$ ← $\Gamma$∪ L(neighbor);
        end
    end
end
return solutions;
```

**Algorithm 4:** A* with Bounding Conflicts

section describes how bounding conflicts are learned in the $k$-best hybrid estimation problem.

Recall that the hybrid observation function is defined as the probability of receiving the observations given the previous sub-belief and the current mode. In this work, it is computed using a Kalman filter on the system-wide continuous state evolution equations defined by Equations 6 and 7. These equations are solved symbolically from the equations that are activated by the mode assignment.

In general, the expected observations may be dependent on every mode assignment. However, engineered systems, especially large engineered systems, tend to be structured in such a way that a change in one section of the system does not necessarily ripple throughout the rest of the system instantly. For example, in the large ship cooling system, valves are normally closed to isolate the pumps from each other. This same effect can be seen in the small, pedagogical fluid system, as closing one of the valves completely isolates the corresponding load from the rest of the system.

This observation suggests that the key to learning bounding conflicts for the hybrid state estimation problem lies in finding some small, relatively independent section of the system (for a given mode assignment) that results in a predicted observation that is significantly different from the actual observed value. Mathematically, the goal is to find some observation $y_{l,c,t}$ (the $l$'th variable in the observation vector) that has a low likelihood and that likelihood depends on only a subset of the modes.

This is shown in the following equations. First, conditional probabilities are used to factor the $P_O$ term in Equation 17. Then, if $p(y_{l,c,t}|...)$ is conditionally independent of all other modes when given some subset $M$ of the mode variables $\hat{\mathbf{m}}^{(j)}_{M,t}$, $P_O$ can be further simplified to Equation 18.

$$\begin{aligned}
\ln P_O(\mathbf{y}_{c,t}, \hat{\mathbf{x}}^{(i)}_{t-1}, \mathbf{u}_{c,t}) &\triangleq \ln p(\mathbf{y}_{c,t}|\hat{\mathbf{m}}^{(j)}_t, \hat{\mathbf{x}}^{(i)}_{t-1}, \mathbf{u}_{c,t}) && (16)\\
&= \ln p(y_{l,c,t}|\hat{\mathbf{m}}^{(j)}_t, \hat{\mathbf{x}}^{(i)}_{t-1}, \mathbf{u}_{c,t}) + \ln p(\mathbf{y}_{1..l-1:l+1:N,c,t}|y_{l,c,t}, \hat{\mathbf{m}}^{(j)}_t, \hat{\mathbf{x}}^{(i)}_{t-1}, \mathbf{u}_{c,t}) && (17)\\
&= \ln p(y_{l,c,t}|\hat{\mathbf{m}}^{(j)}_{M,t}, \hat{\mathbf{x}}^{(i)}_{t-1}, \mathbf{u}_{c,t}) + \ln p(\mathbf{y}_{1..l-1:l+1:N,c,t}|y_{l,c,t}, \hat{\mathbf{m}}^{(j)}_t, \hat{\mathbf{x}}^{(i)}_{t-1}, \mathbf{u}_{c,t}) && (18)
\end{aligned}$$

If an observation can be found such that the first term in the factorization is very negative and it depends on only a subset of the mode assignments, *any* full mode assignment containing that subset will have a $-\ln P_O$ term that is *at least* $-\ln p(y_{l,c,t}|\hat{\mathbf{m}}^{(j)}_{M,t}, \hat{\mathbf{x}}^{(i)}_{t-1}, \mathbf{u}_{c,t})$. This results in the following bounding conflict, where $f$ is the original bounding function of the problem:

$$\left\langle \hat{\mathbf{x}}^{(i)}_{t-1} \cup \mathbf{m}^{(j)}_{M,t}, b(Y) = f(Y) - \ln p(y_{l,c,t}|\hat{\mathbf{m}}^{(j)}_{M,t}, \hat{\mathbf{x}}^{(i)}_{t-1}, \mathbf{u}_{c,t}) \right\rangle \quad (19)$$

Furthermore, bounding conflicts could be found that result from factoring the hybrid observation function to have multiple observations in the first term. This is especially useful if some subset of the observations depend on the same subset of mode assignments.

In order to determine what mode assignments a particular observation depends on, we perform causal analysis while symbolically solving the continuous dynamics equations (Nayak, 1995; Trave-Massuyes & Pons, 1997). This causal analysis keeps track of which equations are used to solve for which variables, as well as which modes introduced each equation. This results in a graph structure, where paths can be drawn from equations and variables to the other variables that depend on them.

Most continuous state estimators provide some easy way to compute the contribution of each sensor observation to the overall observation likelihood. For instance, in a Kalman filter this can be computed trivially using the residual and residual covariance, as shown in Equation 20. This turns the search for a bounding conflict into a search over the observation variables to determine which one contributes the most to making the overall observation unlikely. Then once it is found, the results from the causal analysis specify which mode assignments contributed to that sensor reading.

$$\ln p(y_{l,c,t}|\hat{\mathbf{m}}_{M,t}^{(j)}, \hat{\mathbf{x}}_{t-1}^{(i)}, \mathbf{u}_{c,t}) = -0.5(\tilde{\mathbf{y}}_{l,t}^{(j)})^{\top}(\mathbf{S}_{l,l,t}^{(j)})^{-1}\tilde{\mathbf{y}}_{l,t}^{(j)} \tag{20}$$

An example of this causal analysis for the pedagogical example is shown in Figure 8. This causal analysis is shown for the pump on, valve 1 open, and valve 2 closed. Each box represents a set of equations that are activated by the assignment to the enclosed mode variables. $\top$ represents a set of equations that is active in every mode. This shows, for example, that the flow rate at point 4 is dependent on the state of valve 1, the pressure at points 6, 2, and 1, and the state of the pump. Notice how everything downstream of valve 2 is independent from everything upstream due to valve 2 being closed. This shows that in any configuration with valve 2 closed, the observation at $y_2$ is independent of every other mode assignment. This fact will be used to learn a bounding conflict in the example execution presented in the next section.

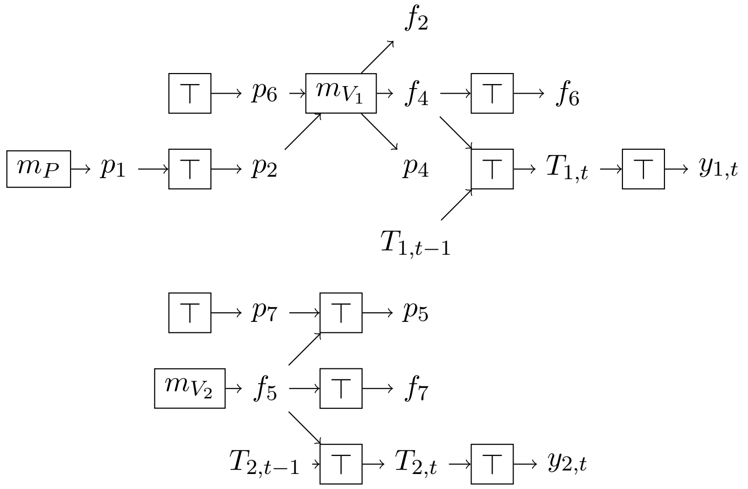
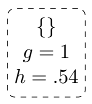


Figure 8: Casual analysis for pedagogical system with the pump on, valve 1 open, and valve 2 closed.

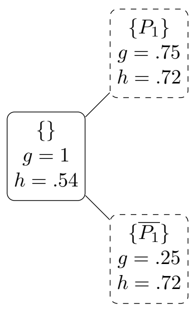
## 6. Example Execution

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 clarity, we will work with probabilities instead of negative ln probabilities (as such, the goal is to maximize instead of minimize).

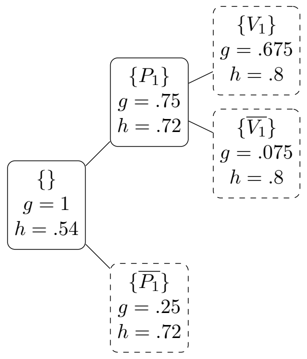
In this example, we will state that the initial mode of the system is known to be $\{P_1, V_1, V_2\}$ with complete certainty and the temperature of each load is described by the Gaussian distribution $\mathcal{N}(400, 1)$. After one time step, the temperature of each load is measured to be 398. Additionally, 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.



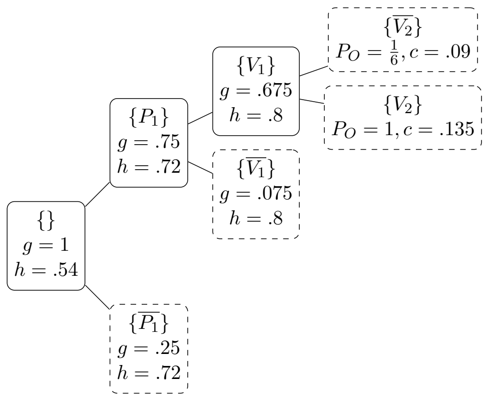
(a) Example execution: step 1.



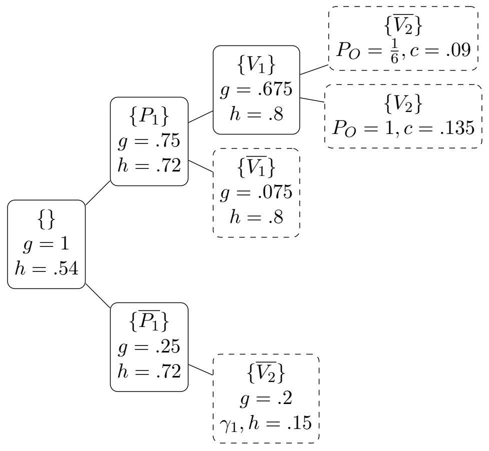
(b) Example execution: step 2.



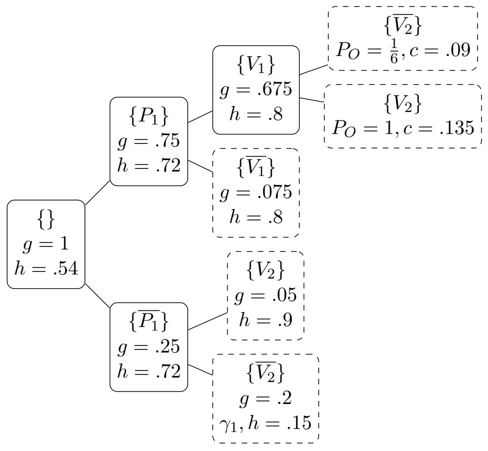
(c) Example execution: step 3.



(d) Example execution: step 4.



(e) Example execution: step 5a.



(f) Example execution: step 5b.

Figure 9: Example execution of A*BC on a $k$-best hybrid state estimation problem.

Table 2: Probability of each component transitioning to a given mode from on/open.

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

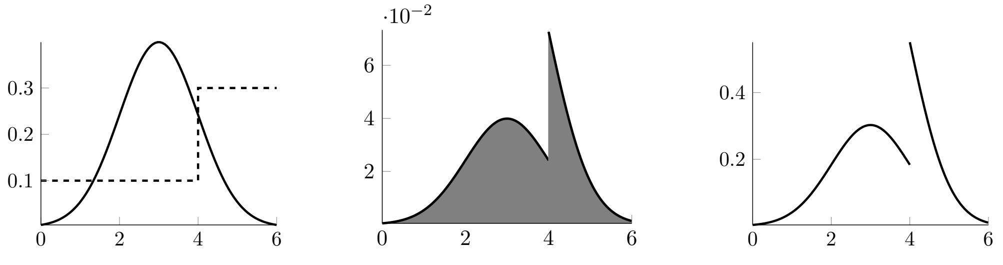
Figure 9 shows the evolution of the search tree over time. Nodes with a dashed outline are the nodes currently on the search queue. The search begins with the root node, representing the empty partial assignment (Figure 9a). No decisions have been made yet, so the $g$ value of the root node is 1 and the $h$ value is 0.54, which comes from multiplying the probability of the most likely transitions for each component. 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 assignments to $P_1$ are first generated (Figure 9b). The $g$ values come from the likelihood of the pump transitioning into each mode and the $h$ values come from maximizing the likelihood of the remaining transitions. The node corresponding to the pump being on is the most likely, so it is extended to include assignments to $V_1$ (Figure 9c) and then the node corresponding to the pump on and valve 1 open is extended to include assignments to $V_2$ (Figure 9d).

At this point there are nodes with full mode assignments. This means a Kalman filter is constructed for these modes, $P_O$ can be computed, and the function $c$ returns the true weight of those modes. 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.

At this point, the conflict learning function provides the newly learned bounding conflict $\gamma_1 = \left\langle \{\overline{V_2}\}, f(x) * \frac{1}{6} \right\rangle$. This comes from the Kalman filter showing that the probability of reading 398 on load 2 is $\frac{1}{6}$ and the causal analysis presented in the previous section showing that observation is dependent only on valve 2 being closed.

The next best search node is the one corresponding to the pump being off. It is popped off the queue and compared to the only known bounding conflict. This search node partially manifests the conflict, meaning that Neighbors-By-BC is used to generate its neighbors. The first neighbor generated manifests the conflict by extending the partial assignment to include that valve 2 is closed. This node is also annotated with the bounding conflict $\gamma_1$ so that a tighter bound can be computed for it. While the standard $h$ function for this node returns 0.9 (because the most likely transition for the unassigned component has probability 0.9), the bounding conflict informs us that the best estimate of $P_O$ for this node is currently $\frac{1}{6}$, not 1. This results in the node having a heuristic value of 0.15 (Figure 9e). Then the remaining neighbors are generated (Figure 9f).



(a) (b) (c)

Figure 10: Interaction between continuous state and mode transitions. In (a), the solid line represents the continuous belief at the previous time ($p_{c,t-1}^{(i)}$) and the dashed line represents the probability of transitioning to the target mode as a function of the continuous state ($\tau$). (b) shows the unnormalized multiplication of $p_{c,t-1}^{(i)}$ and $\tau$. The area under the curve represents the probability of transitioning to the target mode given the previous continuous belief (the value of the hybrid transition function). (c) shows the properly normalized prior to be used by the continuous state estimator.

In the very next iteration of A*BC the node corresponding to $\{P_1, V_1, V_2\}$ will be dequeued and proven to be the optimal mode assignment. If A*BC had not been used, an A* algorithm would have needed to explore the sub tree with the pump off further before being able to prove $\{P_1, V_1, V_2\}$ was the optimal solution (specifically, $\{\overline{P_1}, V_1, \overline{V_2}\}$ would have needed to be placed on the queue first).

## 7. 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.

### 7.1 A Note on Computing the Continuous Prior Distribution

When using Kalman filter variants to perform the continuous filtering there is a subtlety involved in computing the continuous state distribution input at every time step. This subtlety is not discussed by prior work on the development of this hybrid filter, but is important when attempting to implement this hybrid filter[5].

When considering a guarded transition, the prior continuous belief given to the continuous filter is not necessarily $p_{c,t-1}^{(i)}$ — there is an interaction between the guard function and the continuous belief. For example, consider a component where the likelihood of transitioning to a failure mode is non-zero only if the temperature is over a threshold value. If this transition is chosen during the search, the continuous probability mass below that temperature threshold should not be considered (by choosing this transition, the continuous prior has been effectively conditioned on the guard being true). This is demonstrated on a

5. This discussion was postponed to the results section as it is not a core contribution of the paper, but is necessary to accurately replicate results.

less extreme example in Figure 10. The true continuous prior given to the continuous estimator is the normalized multiplication of $p_{c,t-1}^{(i)}$ and the transition likelihood as a function of the state.

Critically, this means that the priors used as input to the one-step continuous state estimator are not always Gaussian in the presence of guarded transitions. To address this, this work uses a truncated unscented Kalman filter (Teixeira, Tôrres, Aguirre, & Bernstein, 2010; Simon, 2010; Garcia-Fernandez, Morelande, & Grajal, 2012) as the continuous estimator and restricts itself to piecewise constant transition probabilities. This means that the prior is broken into regions where $\tau$ 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 for an unscented Kalman filter. This approach results in a single sub-belief and target mode pair having several children sub-beliefs, each with a different continuous belief.

### 7.2 Testing Environment

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 6 and 7. A Fibonacci heap (Fredman & Tarjan, 1987) was used to maintain A*BC's open queue, the Neighbors-By-Variable algorithm chose variables in order of increasing domain size, and the bounding conflict used by Neighbors-By-BC was always chosen to be the bounding conflict with the smallest partial assignment that the search node potentially manifested.

In all experiments, a bounding conflict was learned by searching for the observation that was furthest from its predicted value (also taking into account sensor noise characteristics), and using causal analysis to determine which mode assignments were responsible for the equations used to determine the predicted value. If the likelihood if the worst single observation was higher than 0.5, no bounding conflict was learned. This is a tunable parameter and 0.5 was chosen to allow the search algorithm to learn areas of the state space that were particularly bad, but keep the number of bounding conflicts learned to a reasonable number.

### 7.3 Simple Automaton

This example comes originally from Section V.A of Hofbaur and Williams (2004). It consists of three automata 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 3a, 3b, and 3c. The transition diagrams for the PHAs are show in Figure 11

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. 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 time steps of the three PHA system are shown in Table 4. The $k$-best runtime column shows the normalized time needed to process all time steps with $k = 20$ and A* search as

Table 3: PHAs for simple experimental CPHA.

(a) Algebraic and difference equations for PHA 1.

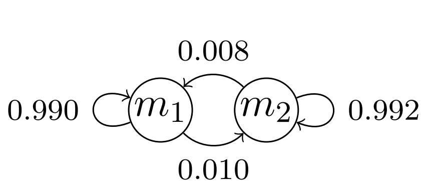
| Mode | Equations |
|---|---|
| $m_{11}$ | $u_{c1} = 2.0w_{c1}$ |
| $m_{12}$ | $u_{c1} = -2.0w_{c1}$ |

(b) Algebraic and difference equations for PHA 2.

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

(c) Algebraic and difference equations for PHA 3.

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



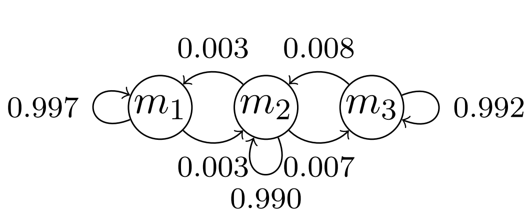


Figure 11: Simple concurrent automata transitions. The left diagram shows the transition probabilities for PHA 1. The right diagram shows the transition probabilities for PHAs 2 and 3.

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 time step, along with the total fraction of time spent in the Kalman filtering routine (including both derivation and execution).

Table 4: Simple Concurrent Automata Results

| $k$ | Search | KF Calls | | | Run Times | |
|---|---|---|---|---|---|---|
| | 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.

### 7.4 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 1a. 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 in this experiment was on understanding the behavior of each method over time on a single time step of the estimator. Once every mode is assigned for a new sub-belief at the next time step, the derivation of the Kalman filtering equations requires solving hundreds of algebraic and difference equations. This was observed to stress Weyl and cause the wall-time for this experiment to be much longer than the previous experiment. Approximately 93% of the total runtime for every run of the system was spent deriving and executing the Kalman filter (with approximately 95% of that time spent in the symbolic solver to derive the filter on demand).

As improving Weyl was not in scope for this research, all results for this benchmark use a timeout of fifteen minutes and use the number of Kalman filter executions as a proxy for time. In each run, approximately 900 Kalman filters were derived and executed —

resulting in the system taking approximately one second to compute a candidate target mode, derive the Kalman filter, execute it, and compute the observation likelihood/learn bounding conflicts.

Figures 12 and 13 show two typical runs of the system with randomly simulated transitions and observations. The upper left graph in each figure shows the number of sub-beliefs proven optimal for time step $t+1$ vs the number of Kalman filter executions. The top right graph shows the unnormalized probability mass of the sub-beliefs produced by the estimator vs the number of Kalman filter executions (the sum of the weights on every sub-belief that is proven optimal). The bottom left graph shows the value of $-\ln(P_o(sn))$ for the full mode assignment tested at each run of the Kalman filter. This represents the error between the bounding function and the true cost of the search node for the A* powered search. The bottom right graph shows the error between the bounding and cost functions using A*BC.

Note that in both of these examples, the A*BC powered version of the estimator produces sub-beliefs covering approximately the same probability mass as the A* powered version an order of magnitude faster (most clearly: the large jump at ∼30-40 Kalman filter invocations vs. ∼300 ). This is due to A*BC learning a better bounding function as the search progresses, as evidenced by the overall lower errors between the bound and cost functions in A*BC vs A*. Note how the error for A*BC starts fairly large (with the same errors as A*), but quickly reduces the magnitude of the error as bounding conflicts are learned. The error in A*BC 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.

To fully compare an A*BC powered $k$-best estimator with an A* powered one, we also include a run of the system where all the noise injected by the environment was close to zero in Figure 14. This resulted in higher observation likelihoods across the board, as can be seen most clearly in the accumulated probability mass graph which is much higher in magnitude than the previous examples. In this situation, there is very little additional information A*BC can learn, so its performance approaches that of A*. However, it is still able to prove some solutions optimal faster than A* can so it remains strictly better in terms of Kalman filter calls.

## 8. 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 enumeration algorithm based on bounding conflicts (A*BC) and have described a state estimator for hybrid discrete and continuous systems built on top of this enumerator.

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

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.

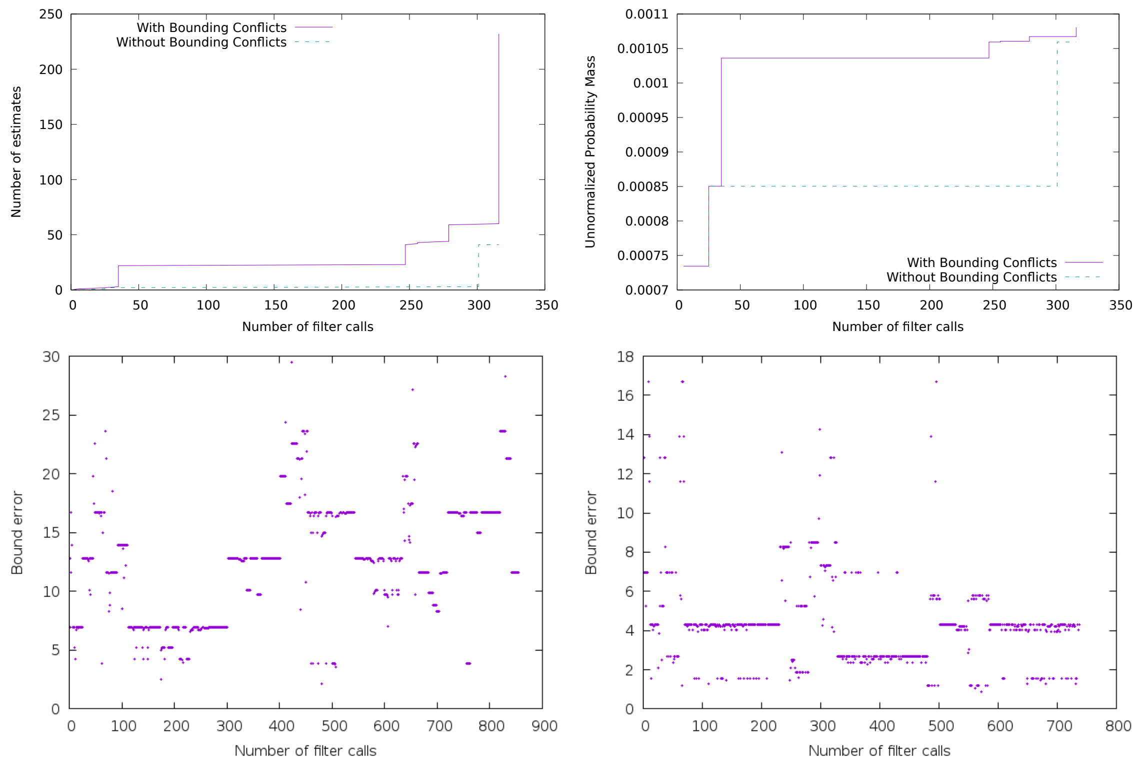


Figure 12: Set one of typical $k$-best filter performance on fluid system benchmark. The upper left graph shows the number of sub-beliefs proven optimal per filter invocation with and without bounding conflicts. The upper right graph shows the unnormalized probability mass covered per filter invocation with and without bounding conflicts. The bottom two graphs show the difference between the estimated observation log-likelihood used in the search heuristic and the actual observation log-likelihood computed by the filter, with the left graph showing the case without bounding conflicts and the right showing with bounding conflicts.

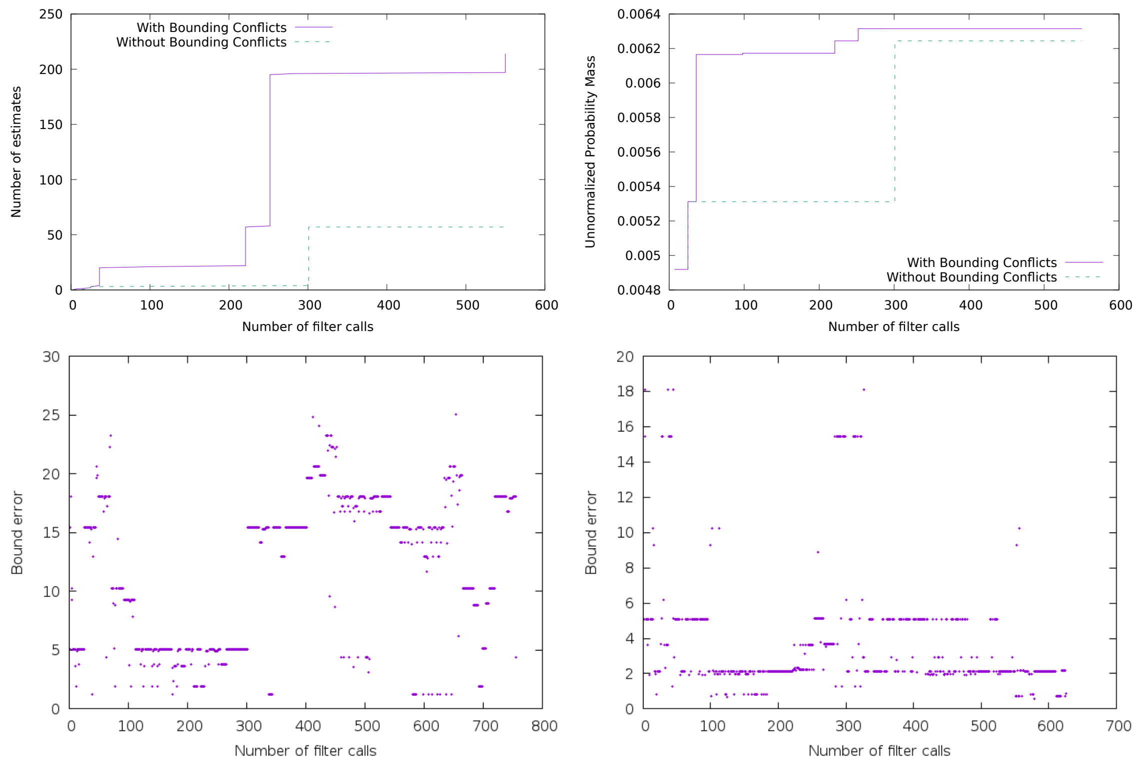


Figure 13: Set two of typical $k$-best filter performance on fluid system benchmark. The upper left graph shows the number of sub-beliefs proven optimal per filter invocation with and without bounding conflicts. The upper right graph shows the unnormalized probability mass covered per filter invocation with and without bounding conflicts. The bottom two graphs show the difference between the estimated observation log-likelihood used in the search heuristic and the actual observation log-likelihood computed by the filter, with the left graph showing the case without bounding conflicts and the right showing with bounding conflicts.

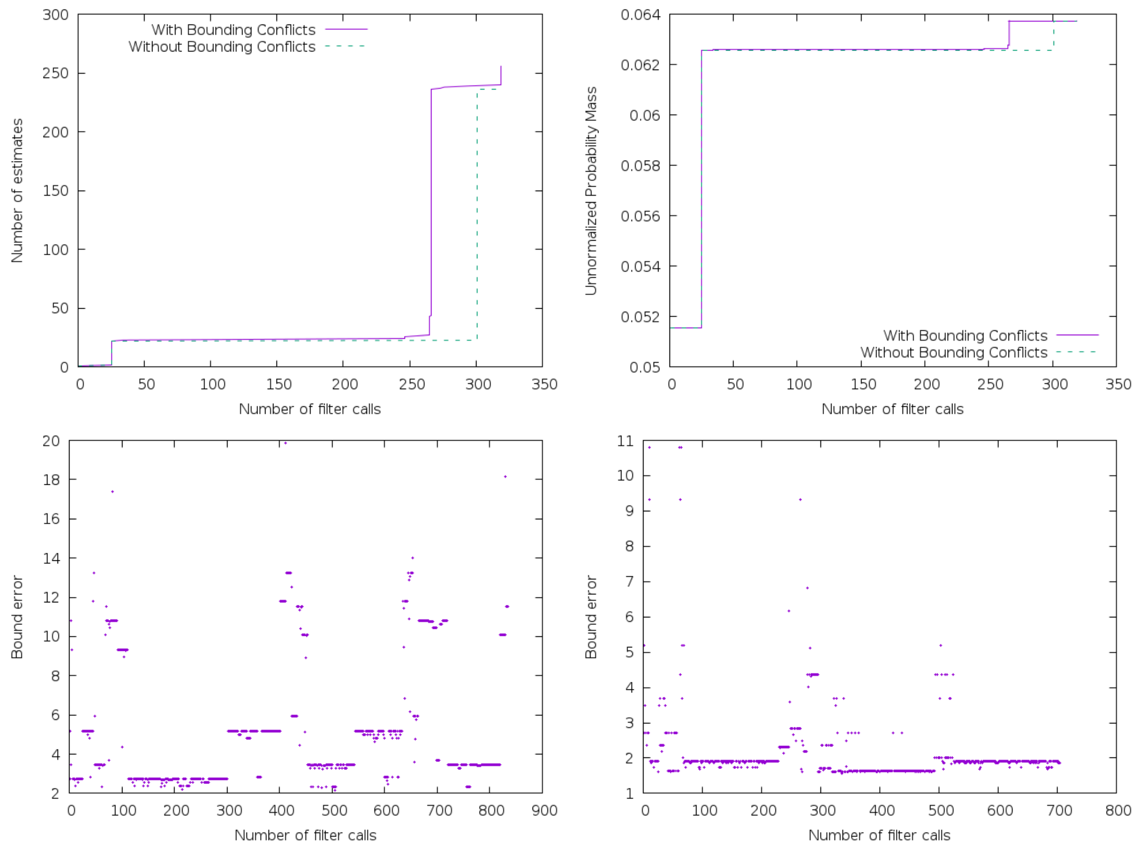


Figure 14: $k$-best filter performance on fluid system benchmark with near-zero noise. The upper left graph shows the number of sub-beliefs proven optimal per filter invocation with and without bounding conflicts. The upper right graph shows the unnormalized probability mass covered per filter invocation with and without bounding conflicts. The bottom two graphs show the difference between the estimated observation log-likelihood used in the search heuristic and the actual observation log-likelihood computed by the filter, with the left graph showing the case without bounding conflicts and the right showing with bounding conflicts.

## Acknowledgments

The authors wish to thank the many researchers in the MIT MERS group who provided feedback on this paper, as well as the anonymous reviewers for their invaluable feedback and efforts to ensure the quality of this paper.

This work was funded by the Office of Naval Research through the Johns Hopkins University Applied Physics Laboratory (MIT contract number 960101) and by the NASA Planetary Science and Technology through Analog Research (PSTAR) program (MIT award number 026426-00001).

## 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 *Proceedings Eighth IEEE International Conference on Tools with Artificial Intelligence*, pp. 132–139. IEEE.

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

Doucet, A. (1998). On sequential simulation-based methods for bayesian filtering. Tech. rep..

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

Fredman, M. L., & Tarjan, R. E. (1987). Fibonacci heaps and their uses in improved network optimization algorithms. *Journal of the ACM (JACM)*, *34*(3), 596–615.

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). New extension of the kalman filter to nonlinear systems. In *Signal Processing, Sensor Fusion, and Target Recognition VI*, Vol. 3068, pp. 182–193. International Society for Optics and Photonics.

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.* Springer.

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. (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. In *International Symposium on Design and Implementation of Symbolic Computation Systems*, pp. 303–318. Springer.