Two Solvers of Partially Observable Stochastic Hybrid Systems

Wenjun Zhao*

Mentor: Dr. Kendra Lesser & Prof. Alessandro Abate

University of Science and Technology of China

wj950328@gmail.com

Abstract

We consider the problem of verification of safety specifications for hybrid systems with a controller that has access to partially observations of the state. Two kinds of POMDP Solvers with different goals are studied. First, we get a finite state approximation of the stochastic hybrid system in a uniform grid. Then we can solve the dynamic programs using the existed algorithms. Finally, we provided some numerical examples of the two solvers and a POMDP GUI which will be available in the future.

I. Introduction

The partially observable stochastic hybrid systems are a general class of dynamical systems when the state information is incomplete or with noise. Because of its generality, partially observable stochastic hybrid systems have found applications in many areas.

In the cases, the main goal is to estimate the probability that the state of a stochastic system stays within some certain set in a given time horizon and provide with an optimal policy to control the system. Further, we have a second goal for the system–not only to control the state within the safe set, but also minimize the cost in the process.

As there already exists some methods on discrete time systems, we should first convert the problem to the discrete edition. Here we used the Discrete time Markov Chain with finite states to approximate the original system. For more information, the theory basis is demonstrated in [1].

The material is organized in five sections. After this brief introduction, in Section 2 we describe the methods we used for computation, both the solver with a cost function or without. Section 3 concentrates on the numerical results of some case study. Section 4 can be regarded as a User's Guide to the Graphical User Interface. Finally, Section 5 outlines some further directions of investigation.

II. Methods

2.1 POMDP Solver

The main results of this part is based on framing the original system as a partially observable Markov decision process. We therefore outline the steps briefly and demonstrate how we realized it in MATLAB. For the detailed algorithm, please see [2] and [3].

Algorithm 1 (POMDP Solver)

1. Define the dynamics and parameters.

2. Generate the discrete approximation model of original states and observations: includes generating the represent points and calculating the transition of matrix.

*The material in this paper is what has been done in the summer research project at the University of Oxford.

3. Sample the belief states according to the model.

4. Calculate the value function based on the back iteration of PBVI method.

5. According to the grid desired for plot, for each grid point, select the value function and calculate the safe probability and option of input. The safe probability is calculated via inner product of states and value functions and the option of input is from the value functions selected for the certain represent point.

2.2 Multi-objective POMDP Solver

As the same method above, the difference mainly lies in the value function because the goals are more than one. For more details, please see [4].

III. Case Study

I. Case Study 1: Linear DC Motor

The DC motor case is a typical system with hybrid states. The input is a voltage source, given by u. The states are ω (angular velocity) and i (current through an inductor). The output is only the angular velocity ω (the current is not measured). The state space equations are:

$$\begin{bmatrix} \dot{\omega} \\ \dot{i} \end{bmatrix} = \begin{bmatrix} -\frac{b}{J} & \frac{K}{J} \\ -\frac{K}{L} & -\frac{R}{L} \end{bmatrix} \begin{bmatrix} \omega \\ i \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{L} \end{bmatrix} u$$

And the observation is:

$$y = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} \omega \\ i \end{bmatrix}$$

The parameters are given below:

b	J	K	L	R
10^{-4}	2.5×10^{-4}	5×10^{-2}	3×10^{-4}	0.5

The voltage input u is restricted to the set U =[-0.25,0.25] and the safe set is [4.5,5]×[-0.25,0.25]. However, we also assume there is additive Gaussian noise corrupting the dynamics and output.

Step 1 Transform the continuous time system to discrete time system.

According to the state equations, we can use MATALB function c2d to convert the system to discrete one. And the discrete time equation has the form below:

$$x_{k+1} = f(x_k, u_k) + v_k \quad (1)$$
$$y_{k+1} = h(x_k) + w_k \quad (2)$$

In the equations above, the v_k and w_k are Gaussian noise with normal distribution with mean 0.
As this case is a linear system, the discrete time version we generated is:

$$x_{k+1} = Ax_k + Bu_k + v_k \quad (3)$$
$$y_{k+1} = Cx_k + w_k \quad (4)$$

$$A = \begin{bmatrix} 0.9998 & 0.01842 \\ -0.01535 & 0.8463 \end{bmatrix},$$
$$B = \begin{bmatrix} 0.003155 \\ 0.307 \end{bmatrix}, C = \begin{bmatrix} 1 & 0 \end{bmatrix}$$

Step 2 Discretize the input, the safe set and the observation set.

In order to apply existed methods to the discrete time system, we need to divide the safe set and observation set into pieces. Here we just made uniform grids with given gap. So we can generate a series of points to represent the safe set and observation set. Here we name them simply as X and Y.

According to the equations (3) and (4), we can calculate the probability that x_{k+1} stay in a certain piece when x_k is in a certain piece with certain input u_k. Hence we can get a three-dimensional transition matrix Txx:

$$Txx(i, i', u) = P(x_{k+1} = X_{i'} | x_k = X_i, u_k = u)$$

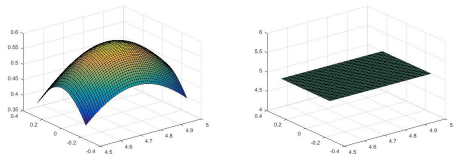
Similarly we can get the two-dimensional matrix Txy to represent the transition probability from X(i) to Y(iplus):

$$Txy(i, i') = P(y_{k+1} = Y_{i'} | x_k = X_i)$$

Here we do those calculations with java to improve the speed.
Step 3 Import the discrete models to the java code already been generated.

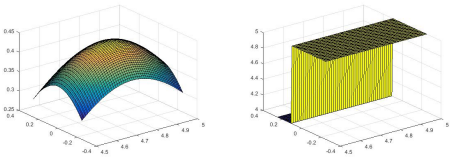
Step 4 Do the calculations with time N to generate the belief states and value functions in java code.

Step 5 For given printing gap, we can calculate the probability for each print point to stay in the safe set and the options of input by finding the value functions. And we can plot them. Here are the results under different noises with different covariances:



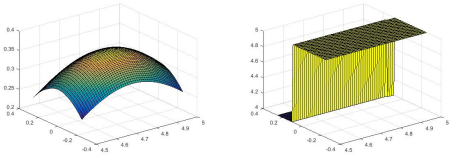
(a) Safe Probability (b) Option of Input

Figure 1: *Covariance of noise = 0.001*



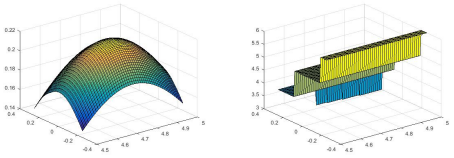
(a) Safe Probability (b) Option of Input

Figure 2: *Covariance of noise = 0.007*



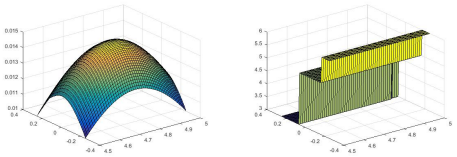
(a) Safe Probability (b) Option of Input

Figure 3: *Covariance of noise = 0.01*



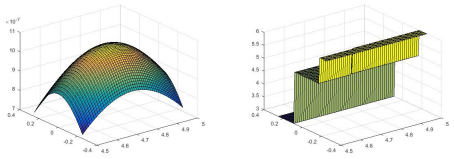
(a) Safe Probability (b) Option of Input

Figure 4: *Covariance of noise = 0.02*



(a) Safe Probability (b) Option of Input

Figure 5: *Covariance of noise = 0.1*



(a) Safe Probability (b) Option of Input

Figure 6: *Covariance of noise = 1*

From the pictures above, we can see that the safe probability is becoming smaller and smaller, and the option of input is also changing.

However, the probability is much lower than we expected. So it is needed to check the safe probability when there is no noise. Codes are also generated to test and the safe probability of every piece equals 1 according to the record. All the plots are generated using our POMDP Solver Graphical User Interface. For more details, please see section 4.

II. Case Study 2: A Heating System

This is a typical example of multi-objective POMDP model. The system has two states:

one is the mode and another is degradation level, a continuous state which can be any value in [0,100]. And the observation also has two states related to each state in the original system.

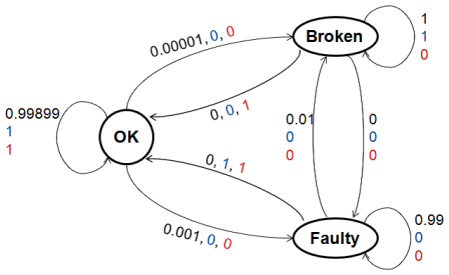


Figure 7: *Mode Transition Probabilities*

The degradation level is modeled as additive process:
OK mode:

$$q = q_1, x_{k+1} = x_k + n_k, n_k \sim |N(\mu, \sigma)|$$

FAULTY mode:

$$q = q_2, x_{k+1} = x_k + c \cdot x_k, n_k \sim |N(\mu, \sigma)|, c > 1$$

BROKEN mode:

$$q = q_3, x_k = 100$$

The observation system is modeled in this way:

$$y_{k+1} = x_k + n_k, n_k \sim |N(0, \sigma)|$$

And the observation model of the modes are also given. The transition matrix is:

$$Tyq = \begin{bmatrix} 0.8 & 0.3 & 0.1 \\ 0.15 & 0.5 & 0.4 \\ 0.05 & 0.2 & 0.5 \end{bmatrix}$$

$$Tyq(y, q) = P(y_2 = y | x_2 = q)$$

There is also three kinds of actions in the system: Do nothing (cost 0), Repair (cost 10) and Replace (cost 1000). The goal is to keep the degradation level in [0,90] while minimize the cost secondarily.

The work we have done is similar in the last case study. The main difference lies in the states. In this case, the mode state only has three possible values. In order to associate the mode state with the continuous state, we first discretize the continuous state [0,100] and then for each represent point, we can relate it to the mode 1,2,3. Thus the number of the discrete approximation of the original system is three times of the discretization of [0,100].

In the following case study, we set the parameters as below:

Discretization level: dx = dy = 2;
Number of belief states: Ns = 100;
Tolerance of Value Function: tol = 0.01;
Time horizon: timeFinal = 30

So when we generate the plots, everything generated have to be regarded as three parts according to different modes. Plots are generated when the time horizon is set to 30:

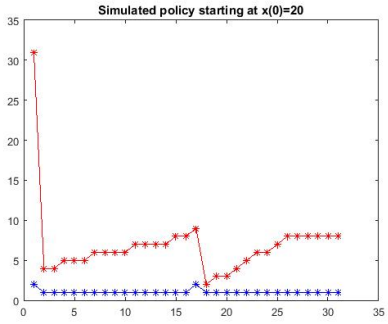


Figure 8: *Simulated Policy at Time Horizon 30*

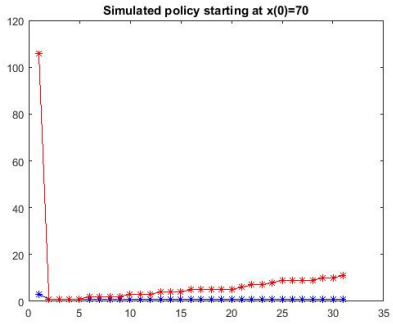


Figure 9: *Simulated Policy at Time Horizon 30*

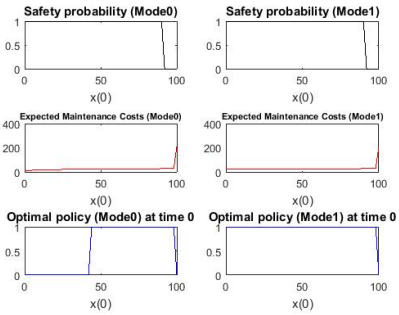


Figure 10: *Safe Probability, Cost and Optimal Policy*

IV. User's Guide of GUI

The graphical user interface of POMDP solver is based on the algorithm of Section 2.1 and the plots in Section 3.1 are generated from this simple GUI.

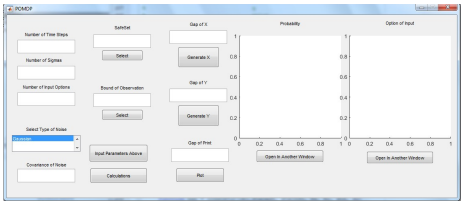


Figure 11: *POMDP Solver*

Then we are going to explain how to use this GUI. According to the case study in Section 3-1, the steps can be done in the GUI.

Step 1 Define the dynamics and the inputs.

The dynamics and inputs are defined in the java class 'MyParameter'. The dynamic equation has the form like this:

$$x_{k+1} = Ax_k + f(u) + v_k$$

And the input u should also be discretized and defined in the java file so we can calculate the finite f(u).

Step 2 Input the parameters needed.
The number of time steps, the number of belief states and the number of input options should be input. These should all be positive integers and the number of input should be larger than 1 (or the solution is trivial). And the covariance of noises (must be positive) should also be input for the calculation of the transition matrix. At present the type of noise should only be Gaussian, further we are going to add other types of noise.

Step 3 Generate the discrete approximation of continuous states and observations.
For the continuous state x, click 'select' to define the safe set according to the guide window. Then input the gap of x with the form of a vector: [gap1,...,gapN]. Then click 'Generate X' to generate the discrete model of X, includes the represent states X and the transition matrix Txx. It is the same to generate the observation Y.

Step 4 Import the models to java.
Click 'Input Parameters Above' to import the discrete model of X and Y.

Step 5 Do the calculations in java.
Click 'Calculations' to calculate the belief states and value functions in java.

Step 6 Generate the plots of safe probability and options of input.
Import the gap of print with the form of a vector. Then click 'plot' to generate the two plots in the axes on the GUI. Click 'Open In Another Window' to get a full view of the plot.

Example in the case study Here is a picture of the GUI to generate the plots in the case study:

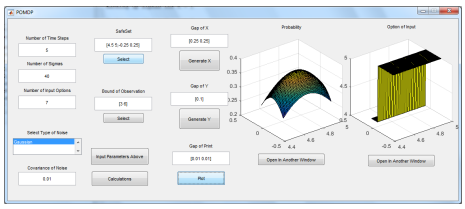


Figure 12: *POMDP Solver*

V. Further Investigation

1. **The efficiency of the algorithm in MATLAB needs to be improved.**

   In the first POMDP Solver, the calculation of transition matrix is done in java to improve the speed. However, in the second one, the transition matrix is calculated in MATLAB to make the import of parameter easier. So the efficiency can be improved to make some adjustment.

   As the number of belief states and value functions are increasing at each iteration, the time of each loop is becoming longer and longer. When the time horizon is very long, it may take much time waiting for the results. Thus it is important to modify the iteration process.

2. **The second POMDP Solver can also be connected in a GUI.**

3. **The GUI has some features to modify.**

   First, the definition of dynamics needs to be defined in java. However, the models are generated in MATLAB and it will be more convenient if it can be defined in MATLAB.

   Second, the type of noise in application may not always be Gaussian. So the listbox on the GUI should add more kinds of noise and then the user can calculate their own transition matrix with different kinds of noise.

References

[1] A.Abate,J.-P.Katoen,J.Lygeros,and M.Prandini. Approximate model checking of stochastic hybrid systems. *European Journal of Control 16*, 6 (2010), 624-641.

[2] K.Lesser,M.Oishi. Finite State Approximation for Verification of Partially Observable Stochastic Hybrid Systems *http://dx.doi.org/10.1145/2728606.2728632.*

[3] K.Lesser,M.Oishi. Reachability for Partially Observable Discrete Time Stochastic Hybrid Systems *Automatica*,50 (2014) 1989-1998

[4] K.H.Wray,S.Zilberstein. Multi-Objective POMDPs with Lexicographic Reward Preferences *http://rbr.cs.umass.edu/shlomo/papers/WZijcai15.pdf.*