Applied Soft Computing 11 (2011) 276–284

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

A framework for the automatic synthesis of hybrid fuzzy/numerical controllers

Daniele Magazzeni

Department of Computer Science, University of L'Aquila, Via Vetoio, Coppito, L'Aquila, Italy

ARTICLE INFO

Article history:
Received 22 December 2008
Received in revised form 19 October 2009
Accepted 16 November 2009
Available online 20 November 2009

Keywords:
Hybrid control
Fuzzy logic
Numerical control
Controller synthesis
Model checking

ABSTRACT

In this paper, a framework for the automatic synthesis of hybrid fuzzy/numerical controllers is proposed. The methodology is based on model checking and on a very precise analysis of a system. This allows one to synthesize optimal numerical controllers and then use them to consistently improve fuzzy controllers. Moreover, we present a new approach that integrates the numerical and the fuzzy components and automatically outputs a hybrid controller. Such a hybrid controller exploits the optimality of numerical controllers and the robustness of fuzzy ones, and it is very compact and fast to read thanks to the use of OBDDs. We apply our methodology to two benchmark problems, the dc motor and the inverted pendulum. The results show that the hybrid controller can handle linear as well as nonlinear systems outperforming both the numerical and the fuzzy controllers.

1. Introduction

Soft computing refers to the integration of artificial intelligence techniques in hybrid frameworks for solving real world problems [1]. In particular, much work is being done to combine different control paradigms in order to obtain *hybrid controllers* with better performance. For example, in [2], neural networks and dynamic programming are used to control a nonlinear process. In this context, the use of *fuzzy logic* together with other approaches has been extensively worked out. For example, in [3] hybrid controllers making use of fuzzy logic and neural networks are proposed, while in [4] genetic algorithms are used to generate and/or calibrate fuzzy controllers. Indeed, fuzzy control represents a powerful technique to cope with continuous systems.

In this paper, we focus on the integration of fuzzy and numerical control, starting from previous works based on *cell mapping*. Cell mapping [5] is based on the discretization of state variables of the system, partitioned into cells, and represents a very efficient computational technique for the global analysis of continuous systems.

Hsu used cell mapping to generate a numerical controller, that is a control table containing state-action pairs [6]. However, the values in the table are discretized and this can introduce steady state errors when dealing with continuous systems. Furthermore, even using a very fine discretization, the table can require a lot of memory for a large number of cells. In contrast, a fuzzy logic controller is continuous and only requires memory for the input fuzzy set definitions and the output function parameters.

Much work is being done to provide methodologies for the automatic calibration of fuzzy logic controllers in order to improve their performance. In particular, cell mapping has been used to generate control tables to improve the quality of fuzzy controllers [7–9].

However, to the best of our knowledge, no previous works have addressed the problem of *simultaneously* using a fuzzy controller together with an optimal control table. In this context, a very important issue is to provide a methodology for the *automatic* generation of such a hybrid controller.

1.1. Our contribution

In this paper, we propose a new approach based on the following considerations:

1. *Model checking* techniques [10,11] allow one to explore huge state spaces (also up to billions of states) and to use a very fine discretization of the state space, so obtaining a more precise analysis of a system than the one provided by other techniques as cell mapping.
2. Recently, these techniques have been used to automatically synthesize *optimal* control tables.
3. In previous works, we proposed an automatic compression process for numerical controllers that uses *ordered binary decision diagrams* (OBDD). The OBDD representation results very compact and very fast to read.
4. Fuzzy controllers still remain very suitable to cope with continuous systems due to their *robustness*.

E-mail address: daniele.magazzeni@univaq.it.

1568-4946/$ – see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.asoc.2009.11.018

We propose to exploit properties of numerical and fuzzy controllers highlighted in 1–4 in order to *automatically* generate a *hybrid controller* consisting of

- *a numerical component:* that is the optimal control table synthesized by model-checking-based tools;
- *a fuzzy component:* that is a fuzzy controller generated and/or calibrated using the information in the numerical controller.

As explained in Section 4, the hybrid controller takes advantage of the optimality of numerical controllers and the robustness of fuzzy ones. Finally, making use of the OBDD-compression, the resulting hybrid controller requires a small amount of memory and can be easily implemented in a microprocessor.

The paper is organized as follows. In Sections 2 and 3 we introduce fuzzy and numerical controllers, respectively. In Section 4 we describe, step by step, our methodology for the automatic generation of hybrid controllers. Section 5 shows experimental results and presents a comparison between the controllers performance. Section 6 concludes the paper.

2. Fuzzy control

Fuzzy control is well known as a powerful technique for designing and realizing controllers, especially suitable when a mathematical model is lacking or is too complex to allow an analytical treatment [12].

A fuzzy controller (FC) is based on qualitative fuzzy rules which have the form "**if** *condition* **then** *control action*", where the *condition* represents the rule premise and the *control action* the rule consequent.

Fuzzy controllers are very effective in handling "uncertain" or "partially known" situations, and therefore may be used to build very robust controllers. Moreover, FCs are well suited to low-cost implementations based on small devices, since the fuzzy knowledge representation is usually very compact. The design of a FC is usually accomplished by "translating" into fuzzy sets and inference rules the knowledge derived from human experts or mathematical models. Moreover, different approaches have been studied to automatically generate a FC by analyzing the plant specifications and/or behavior. As shown in [13], a FC can also be generated from a numerical controller. These approaches are all based on some kind of automatic statistical analysis, which can be done using various techniques, including neural networks and genetic algorithms [14,15].

In the following, we focus on *Takagi–Sugeno* FC, widely used in the context of automatic generation of FC.

2.1. Takagi–Sugeno fuzzy controllers

The main feature of a Takagi–Sugeno (TS) fuzzy controller [16] is that the rule consequents are linear functions of the input variables, while in traditional fuzzy controllers they are fuzzy sets.

For example, a simple TS fuzzy controller relating two inputs, *error* and *change in error*, to the output *action*, might consist of n rules of the following form:

$$rule[j]: \begin{cases} \text{IF} & \text{error}\,(x_1)\,\text{is Negative} \\ \text{AND} & \text{change in error}\,(x_2)\,\text{is Zero} \\ \text{THEN} & \text{action}\,(u_j)\,\text{is}\,c_{j1}x_1 + c_{j2}x_2 + c_{j3} \end{cases}$$

Thus, let d_j be the degree of truth of the jth rule, the weighted average of the outputs $u_1, \ldots, u_n$ is used to compute the final output of the TS fuzzy controller:

$$u = \frac{\sum_{j=1}^{n} d_j u_j}{\sum_{j=1}^{n} d_j}$$

Obviously, the performance of a TS fuzzy controller strongly depends on the coefficients c_j and much work is being done to provide (semi)-automatic methods for improving such coefficients [7–9].

2.2. Cell mapping

Cell mapping allows an approximated analysis of a state space [5]. The approximation stems from the fact that the continuous state space is partitioned into a finite number of disjoint cells. Thus, each variable ranges on the set of cells, instead of $\mathbb{R}^n$. The main effect of cell partition is that all elements in a cell are approximated with the center point of that cell.

The cell map of a system is constructed using an *unravelling algorithm* to compute cell trajectories. Based on these trajectories, one can establish which cells converge to the set point (controllable cells) and which not (uncontrollable cells).

We refer the reader to [5] for further details about cell mapping, the unravelling algorithm and applications. Here we only stress that cell mapping requires a *global* analysis of the state space, and thus, for complex systems, when a high precision is required, cell mapping is *hard to apply*.

3. Numerical controllers

As mentioned above, a numerical controller is a table, indexed by plant states, whose entries are commands for the plant. These commands are used to set the control variables in order to reach the set point from the corresponding states. Namely, when the controller reads a state from the plant, it looks up the action described in the associated table entry and sends it to the plant.

There is a number of well-established techniques for the synthesis of numerical controllers. Among them, one of the most versatile and widely used technique is *dynamic programming*, which is very suitable for the generation of optimal controllers [17]. However, the dynamic programming approach often requires the definition of design functions which have to be chosen case by case and the inversion of the dynamical behavior of the system which can be hard to compute.

Since in this paper we are interested in automatic methodologies for the controller synthesis, we consider an alternative approach based on model checking.

3.1. Model checking for the generation of optimal numerical controllers

Model checking techniques [10,11] allow one to automatically synthesize optimal controllers starting from the plant description [18]. The main idea is that, in order to build a controller for a plant $\mathcal{P}$, a suitable discretization of the state space of $\mathcal{P}$ is considered, as well as of the control actions u. The plant behavior, under the (discretized) control actions, gives rise to a transition graph $\mathcal{G}$, where the nodes are the reachable states, and a transition between two nodes models an allowed control action between the corresponding states. In this setting, the problem of designing the optimal controller reduces to finding the minimum path in $\mathcal{G}$ between each state and the nearest *goal* state.