Neurocomputing 150 (2015) 116–123

Contents lists available at ScienceDirect

Neurocomputing

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

A L-MCRS dynamics approximation by ELM for Reinforcement Learning

Jose Manuel Lopez-Guede [a,d,*], Borja Fernandez-Gauna [b,d], Jose Antonio Ramos-Hernanz [c]

[a] Department of Systems Engineering and Automatic Control, University College of Engineering of Vitoria, Basque Country University (UPV/EHU), Nieves Cano 12, 01006 Vitoria, Spain
[b] Department of Software and Computing Systems, University College of Engineering of Vitoria, Basque Country University (UPV/EHU), Nieves Cano 12, 01006 Vitoria, Spain
[c] Department of Electrical Engineering, University College of Engineering of Vitoria, Basque Country University (UPV/EHU), Nieves Cano 12, 01006 Vitoria, Spain
[d] Computational Intelligence Group, Basque Country University (UPV/EHU), Spain

ARTICLE INFO

Article history:
Received 29 October 2013
Received in revised form
6 January 2014
Accepted 29 January 2014
Available online 2 October 2014

Keywords:
Extreme learning machines
Linked multicomponent robotic systems
Hose control
Reinforcement learning

ABSTRACT

Autonomous task learning for Linked Multicomponent Robotic Systems (L-MCRS) is an open research issue. Pilot studies applying Reinforcement Learning (RL) on Single Robot Hose Transport (SRHT) task need extensive simulations of the L-MCRS involved in the task. The Geometrically Exact Dynamic Spline (GEDS) simulator used for the accurate simulation of the dynamics of the overall system is a time expensive process, so that it is infeasible to carry out extensive learning experiments based on it. In this paper we address the problem of learning the dynamics of the L-MCRS encapsulated on the GEDS simulator using an Extreme Learning Machine (ELM) approach. Profiting from the adaptability and flexibility of the ELMs, we have formalized the problem of learning the hose geometry as a multi-variate regression problem. Empirical evaluation of this strategy achieves remarkable accurate approximation results.

1. Introduction

A relevant taxonomy of different types of Multi-Component Robotic Systems was introduced in [1]. That taxonomy was structured focusing on the degree of coupling among the robots composing the system, characterized by the strength of their coupling. The first type of systems is those where there is no physical coupling, namely Uncoupled systems. Examples of this kind of systems are robot soccer teams, teams of unmanned aerial vehicles (UAV) and uncoupled swarms. A second type of systems arises when a rigid physical coupling is considered between the robotic components, generating then a new unit with new physical and functional properties. This type of systems is called Modular systems. Some examples are PolyBot, M-TRAN, Proteo, some cases of S-bots and industrial robots properly coupled. The last type of systems is composed of elements coupled through a passive non-rigid element, and is called Linked systems. Being more specific in the discussion on the third type of systems, Linked Multi-Component Robotic Systems (L-MCRS) [1] are autonomous robot groups attached to a non-rigid unidimensional object linking them, which imposes constraints in the robot dynamics which interfere in their coordination, introducing strong non-linearities in the dynamics of the system, and consequent uncertainty in the control of the robots. It is also a non-linear transmission medium for the dynamical influences among robots. With regard to the applications in which this kind on systems can play an important role, most of them are related to some function the non-rigid link itself. The paradigm is illustrated by the transportation of a hose-like object [2,3], more specifically the transportation of the hose tip to a given location in the working space while the other extreme is attached to a fixed point. The paradigmatic task is the Single Robot Hose Transport (SRHT). Achieving SRHT control can be generalized to more complex tasks. These complex tasks include the transport of liquids as water (for supply, for fire fighting or for smart orchards), paint, fuel, etc., the transport of electrical energy or even air (compressed or not), among others. They can also perform tasks of collecting liquids as water in floodings or fuel, oil and toxic substances in accidents, or of collecting semisolids as garbage. To simulate the dynamics of this kind of systems and the effect of a robot action on its components (mainly on the hose), we use an accurate hose dynamics model based on Geometrically Exact Dynamic Splines (GEDS) [2], which is computationally expensive.

Autonomous learning of control tasks in L-MCRS is still an open research issue. We have used Reinforcement Learning (RL) techniques to deal with it, modeling the problem as a Markov Decision Problem (MDP). There are several RL strategies, but when a model-free strategy is chosen, learning performs the repetition of the experiments a large number of times. In many domains, this is not a problem by itself, however, in the domain of L-MCRS where the experimentation is carried out by means of a computationally

* Corresponding author.
E-mail address: jm.lopez@ehu.es (J.M. Lopez-Guede).

http://dx.doi.org/10.1016/j.neucom.2014.01.076
0925-2312/

expensive simulator, the realization of the reported experimental results [3–7] consumes long times.

To solve this issue, learning approximations to the GEDS model by Artificial Neural Networks (ANN) have been proposed [8]. A trained ANN model provides very fast responses allowing us to perform exhaustive simulations for RL. In this paper we focus on the development of an Extreme Learning Machines (ELM) approximation to the GEDS model. ELMs are very adaptable and flexible neural networks, and besides, they have several interesting properties: their training is very fast, they are easy to implement and need minimal human intervention, allowing the formulation of the GEDS model learning as multi-variable regression task. We analyze the accuracy of the approximation to the GEDS model, concluding that the tradeoff between the quality of the approximation and its response speed allows further use of the approximate model embedded in RL experiments.

The paper is structured as follows. Section 2 recalls the definitions of some computational methods involved in the developed work. Section 3 introduces the problem that we are addressing in the paper, while Section 4 presents the ELM based approach to solve it. Section 5 details the experimental setup that has been carried out, and Section 6 discusses the obtained experimental results. Finally, Section 7 presents out conclusions and addresses future work.

2. Computational methods

In this section we review computational methods used along the paper which are essential to understand the approximation problem that we are facing, its magnitude, and the solutions that we are reporting. Section 2.1 gives a review on Geometrically Exact Dynamic Splines (GEDS) to understand how the unidimensional element of the L-MCRS is modeled. Section 2.2 reviews some basic concepts of Reinforcement Learning (RL), specifically the Q-Learning (Q-L) and TRQ-Learning (TRQ-L) algorithms. Finally, Section 2.3 gives a short review of Extreme Learning Machines (ELMs).

2.1. Geometrically exact dynamic splines

We assume a simplified L-MCRS model where the hose is a one-dimensional object deployed in the two dimensional space of the ground. The basic geometrical model of the hose is a spline [9], that is, a linear combination of control points $\mathbf{p}_i$ where the linear coefficients are the polynomials $N_i(u)$ which depend on the length of the curve parameter u defined in $[0,1)$. Formally: $\mathbf{q}(u)=\sum_{i=0}^{n} N_i(u)\cdot\mathbf{p}_i$, where $N_i(u)$ is the polynomial associated to the control point $\mathbf{p}_i$, $\mathbf{q}(u)$ is the point of the curve at the parameter value u. It is possible to travel over the curve by varying the value of parameter u, starting at one end for $u=0$, finishing at the other end for $u=1$. We need to specify some specific positions over the curve corresponding to the robot positions. Therefore, a more appropriate model is a B-Spline defined as follows. Given $n+1$ control points $\{\mathbf{p}_0,\ldots,\mathbf{p}_n\}$ and a knots' vector $\mathbf{U}=\{u_0,\ldots,u_m\}$, the B-spline curve of degree p is $\mathbf{q}(u)=\sum_{i=0}^{n} N_{i,p}(u)\cdot\mathbf{p}_i$, where $N_{i,p}(u)$ are B-spline basis functions of degree p ($p=3$ in this work), built using the Cox–de-Boor's algorithm [10]. Furthermore, we expect that our system will be changing in time, so the appropriate model is a dynamic B-spline whose control points depend on the time parameter t, that is $\mathbf{q}(u,t)=\sum_{i=0}^{n} N_{i,p}(u)\cdot\mathbf{p}_i(t)$.

In hose transport systems, either by a single robot or a team of robots, the dynamics of the B-spline control points are determined by the forces exerted by the robots and the intrinsic forces acting on the hose: stretching, bending, inertia, friction, and twisting moment. The dynamical model for the hose simulation, based on the GEDS approach [11] and Cosserat rod approach [12], is detailed in [2,3,13]. The simulation of the effect of the robot control commands on the hose shape is computed using the following linear local approximation model [2]: $\mathbf{F}_p=J_{pr}\cdot\mathbf{F}_r$, where the relation between the forces applied on the robot attaching points $\mathbf{F}_r$ and the resulting forces on the spline control points $\mathbf{F}_p$ are given by the Jacobian matrix J_{pr} relating robot positions and control points:

$$J_{pr}=\begin{pmatrix} \frac{\partial\mathbf{q}(u_{r_1})}{\partial\mathbf{p}_0} & \cdots & \frac{\partial\mathbf{q}(u_{r_l})}{\partial\mathbf{p}_0} \\ \vdots & \ddots & \vdots \\ \frac{\partial\mathbf{q}(u_{r_1})}{\partial\mathbf{p}_n} & \cdots & \frac{\partial\mathbf{q}(u_{r_l})}{\partial\mathbf{p}_n} \end{pmatrix}=\begin{pmatrix} N_0(u_{r_1}) & \cdots & N_0(u_{r_l}) \\ \vdots & \ddots & \vdots \\ N_n(u_{r_1}) & \cdots & N_n(u_{r_l}) \end{pmatrix}, \quad (1)$$

where the robot positions u_{r_i} correspond to the B-spline knots.

2.2. Reinforcement learning

Reinforcement Learning (RL) [14] is a class of learning algorithms which assumes that the environment-agent system can be modeled as a discrete time stochastic process formalized as a Markov Decision Process (MDP) [15,16], defined by the tuple $\langle S, A, T, R\rangle$, where S is the state space, A is the action repertoire, specifically A_s are the actions allowed in state $s\in S$, $T: S\times A_s\times S\to\mathbb{R}$ is the probabilistic state transition function, and $R: S\times A_s\to\mathbb{R}$ is the immediate reward function. A policy $\pi: S\to A_s$ is the probabilistic decision of the action $a\in A_s$ to be taken in state $s\in S$. RL procedures look for optimal action selection policies maximizing the total reward received by the agent.

2.2.1. Q-Learning

Algorithm 1. Q-Learning algorithm.

Initialize $Q(s,\ a)$ arbitrarily
Repeat (for each episode):
  Initialize s
  Repeat (for each step of episode):
    Choose a from s using policy derived from Q
    Take action a, observe reward r and new state s'
    $Q(s,\ a)\leftarrow Q(s,\ a)+\alpha\left[r+\gamma\max_{a'} Q(s',\ a')-Q(s,\ a)\right]$
    $s\leftarrow s'$
  until s is terminal

Q-Learning [17] is an unsupervised model free RL algorithm that learns the optimal policy in environments specified by Finite MDP (S and A are finite sets). The learning process is specified in Algorithm 1. The main idea of the algorithm is to fill a lookup table $Q(s,a)$ of dimensions $|S|\times|A|$, which is initialized arbitrarily, being updated following the rule specified by the following equation:

$$Q(s_t,\ a_t)\leftarrow Q(s_t,\ a_t)+\alpha\left[r_{t+1}+\gamma\cdot\max_{a} Q(s_{t+1},\ a)-Q(s_t,\ a_t)\right]. \quad (2)$$

It has been proved [18] that, for a discrete FMDP environment, the Q-Learning algorithm converges with probability one to the optimal policy if α decrease complies with the stochastic gradient convergence conditions and if all actions are infinitely sampled in all states.

2.2.2. TRQ-learning

Algorithm 2. TRQ-Learning algorithm.

Initialize $Q(s,a)$ with arbitrary random values
Initialize $T(s,a)=$ ï¿œ, $R(s,a)=0$ for all states $s\in S$ and actions $a\in A$.
Repeat (for each episode):
  Initialize s