Ipseity – A Laboratory for Synthesizing and Validating Artificial Cognitive Systems in Multi-Agent Systems

Fabrice Lauri, Nicolas Gaud, Stéphane Galland, and Vincent Hilaire

IRTES-SET, UTBM, 90010 Belfort cedex, France
{fabrice.lauri,nicolas.gaud,stephane.galland,vincent.hilaire}@utbm.fr

Abstract. This article presents an overview on IPSEITY[1], an open-source rich-client platform developed in C++ with the *Qt*[2] framework. IPSEITY facilitates the synthesis of artificial cognitive systems in multi-agent systems. The current version of the platform includes a set of plugins based on the classical reinforcement learning techniques like Q-Learning and Sarsa. IPSEITY is targeted at a broad range of users interested in artificial intelligence in general, including industrial practitioners, as well as machine learning researchers, students and teachers. It is daily used as a course support in Artificial Intelligence and Reinforcement Learning and it has been used successfully to manage power flows in simulated microgrids using multi-agent reinforcement learning [2].

Keywords: Multi-Agent Systems, Reinforcement Learning.

1 Introduction

Multi-agent systems constitute a fitted paradigm for solving various kinds of problems in a distributed way or for simulating complex phenomena emerging from the interactions of several autonomous entities. A multi-agent system (MAS) consists of a collection of *agents* that interact within a common environment. Every agent perceives some information extracted from its environment and acts upon it based on these perceptions.

The individual behaviors of the agents composing a MAS can be defined by using many decision making mechanisms and many programming languages according to the objective at hand. For instance, a planification mechanism can be used to fulfill the agent goals like in Jason[3]. A powerful alternative is to implement Reinforcement Learning (RL) algorithms that allow the agents to learn how to behave rationally. In this context, a learning agent tries to achieve a given task by continuously interacting with its environment. At each time step, the agent perceives the environment state and performs an action, which causes

[1] `http://www.ipseity-project.com`
[2] `http://www.qt-project.org`
[3] `http://jason.sourceforge.net/wp/`

the environment to transit to a new state. A scalar reward evaluates the quality of each transition, allowing the agent to observe the cumulative reward along sequences of interactions. By trials and errors, such agents can manage to find a policy, that is a mapping from states to actions, which maximizes the cumulative reward.

To our knowledge, there is currently no multi-agent platform that allow users interested in multi-agent RL in particular to easily study the influence of some computational assumptions on the performance and the results obtained by different dedicated algorithms using accepted benchmarks. Indeed, Weka [4] is dedicated to supervised and unsupervised machine learning techniques and the RL toolbox by Neumann [3] is dedicated to control tasks involving only one agent.

The above mentioned *computational assumptions* may encompass hypothesis on:

- system scheduling (When the system is updated? Which agents are allowed to perceive at a given time? Which agents are allowed to act at a given time?);
- design of a cognitive system possibly based on RL (What are its constitutive modules?);
- association of the cognitive systems with the set of agents;
- integration of the current percept of an agent to its long-term memory. This long-term memory may represent the state of the reliable information upon which it will base any of its future decisions.

IPSEITY is intended to help studying all these aspects.

2 Overview

IPSEITY is a rich-client platform especially dedicated to facilitating the implementation and the experimental validation of different kinds of behaviors for cooperative or competitive agents in MASs.

2.1 Kernel concepts

In IPSEITY, a set $\mathcal{A}$ of *agents* interact within a given *environment*. A set $\mathcal{G} = \{\mathcal{G}_1, \cdots, \mathcal{G}_N\}$ of agent groups, called *taxons* in IPSEITY, can be defined. Agents grouped together into the same taxon are likely to behave similarly (they share the same decision making mechanism). The behavior of an agent is exhibited according to its *cognitive system*. A cognitive system implements the decision process that allows an agent to carry out actions based on its perceptions. It can be plugged directly to a given agent or to a taxon. In the latter case, all the agents associated to the same taxon use the same decision process. If a cognitive system is based on RL techniques, plugging it to a taxon shared by several agents may speed up their learning in some cases, as any agent can immediately benefit from the experiences of the others.

The agents interact within the environment from different possible initial configurations, or *scenarios*. Scenarios allow the user to study the quality of the decisions taken individually or collectively by the agents under some initial environmental conditions. During a *simulation*, the agents evolve within the environment by considering several predefined scenarios whose order is handled by a *supervisor*, who can be the user himself or an agent.

2.2 Properties

IPSEITY was designed to possess the following properties:

Flexibility: IPSEITY uses kernel concepts and components (i.e. data representations and algorithms) that are as flexible as possible. For example, the perceptions and the responses of agents are represented by 64-bit float vectors, allowing agents to be immerged either in discrete environments or in continuous environments.

Modularity: IPSEITY uses modules, or *plugin*, to implement kernel concepts and components. For example, the environments, the cognitive systems, the agent scheduling, and the selection of the simulation scenarios are all defined as plugins. These plugins (and in particular those related to cognitive systems) can be easily integrated in other systems and applications. For example, IPSEITY can be used to learn the behaviors of some agents. Once the learning phase is finished, agents can perceive information from a remote environment and act according to the learnt behavior. Such integration has been realized between IPSEITY and JANUS [1], where agents with learnt behaviors communicate within a microgrid simulator developed under JANUS.

Extensibility: IPSEITY can easily be extended by specialized plugins for the target application area. Customized extensions include new environments, algorithms that take part in the decision processes of some cognitive systems or rendering modules for some predefined environments.

System analysis: IPSEITY supports the user in keeping track of all the data, performed actions and results of a RL task. This data set is linked to a session, allowing the user to easily and rapidly compare the results obtained from different algorithms when these algorithms have been executed using different sessions.

2.3 Application example

IPSEITY has been used as a simulation platform for testing several Multi-Agent Reinforcement Learning techniques applied to the management of power flows in a microgrid [2]. An environment representing a microgrid consisting of several sources (batteries, supercapacitors, photovoltaic panels, wind generators and a main grid) and loads has been implemented. In this environment, the batteries and the supercapacitors need to be controlled in order to balance supply and demand whilst minimizing the contribution of the main grid. Several scenarios were defined by specifying the number of sources, the number of loads as well

as the profiles of the renewable energy sources and the loads. They were used in the learning phase and the evaluation phase. Each MARL techniques were compared on the basis of the results obtained from the same set of scenarios. For example, the comparison of the techniques mainly focused on studying the influence in sharing or not the Q-value memory among agents of the same taxon to speed up learning. The evaluation criteria concern the total grid contribution of the main grid during the period of the simulation, its average and standard deviation, as well as the stability duration of the microgrid.

3 Selected plugins

The cognitive systems in IPSEITY can be decomposed into several plugins. Each of them takes part in the decision process. For example, the reinforcement learning cognitive system is currently built from three classes of plugins: a *behavior module*, that selects actions from states, a *memory*, that stores the Q-values of the state-action pairs, and a *learning module*, that updates the contents of the memory from data obtained after environmental transitions. Currently, *Epsilon-Greedy* and *Softmax* are predefined plugins that can be used as behavior modules, *Q-Learning* and *Sarsa* have been implemented and can be used as learning modules. The Q-value memory can be instanciated by a plugin implementing either a static or a dynamic lookup table, or a linear function approximator using a *feature extraction module* like CMAC for example. More details about the kernel concepts and about the software architecture are available on the web site.

4 Conclusion

An overview of IPSEITY has been presented in this article, focusing on RL. IPSEITY is highly modular and broadly extensible using plugins. It can be freely downloaded from `http://www.ipseity-project.com` under a GNU *GPLv3* open-source licence. It is intended to be enriched with state-of-the-art RL techniques very soon. Persons who want to contribute to this project are cordially encouraged to contact the first author.

References

1. N. Gaud, S. Galland, V. Hilaire, and A. Koukam. An organisational platform for holonic and multiagent systems. In *Programming Multi-Agent Systems*, volume 5442 of *Lecture Notes in Computer Science*. 2009.
2. F. Lauri, G. Basso, J. Zhu, R. Roche, V. Hilaire, and A. Koukam. Managing Power Flows in Microgrids using Multi-Agent Reinforcement Learning. In *Agent Technologies in Energy Systems (ATES)*, 2013.
3. G. Neumann. *The Reinforcement Learning Toolbox, Reinforcement Learning for Optimal Control Tasks*. PhD thesis, Institt for Grundlagen der Informationsverarbeitung (IGI), 2005.
4. I.H. Witten, F. Eibe, and M.A. Hall. *Data mining: Practical Machine Learning Tools and Techniques (Third edition)*. Morgan Kaufmann, 2011.