# 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 [4].

**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. 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 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 (learning) parameters on the performance and the results obtained by different dedicated algorithms using accepted benchmarks. Indeed, RL-Glue[1], CLSquare[2], PIQLE[3], RL Toolbox[4], JRLF[5], LibPG[6]only support single-agent RL techniques. The MARL Toolbox[7] supports multi-agent reinforcement learning under Matlab, but unlike Ipseity, it is not possible to remotely control other systems.

## 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.

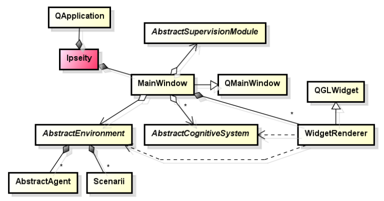
The agents interact within the environment from different possible initial configurations, or *scenarios*. Scenarios allow the user to study the quality of the decisions taken individually or collectively by the agents under some initial environmental conditions. During a *simulation*, the agents evolve 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 Platform Architecture

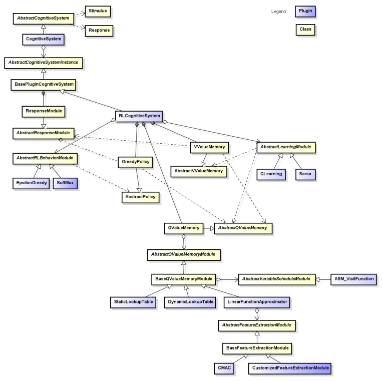
Ipseity simulates discrete-time or continuous-time multi-agent environments inherating from *AbstractEnvironment* (see Fig. 1a). Several agents (inherating from *AbstractAgent*) interact in a multi-agent environment on the basis of their cognitive system inherating from *AbstractCognitiveSystem*. A set of scenarii has to be defined in order to carry out the simulations. These are selected by a *AbstractSupervisionModule*.

[8] `http://glue.rl-community.org/wiki/Main_Page`
[9] `http://ml.informatik.uni-freiburg.de/research/clsquare`
[10] `http://piqle.sourceforge.net`
[11] `http://www.igi.tu-graz.ac.at/gerhard/ril-toolbox/general/overview.html`
[12] `http://mykel.kochenderfer.com/jrlf`
[13] `https://code.google.com/p/libpgrl`
[14] `http://busoniu.net/repository.php`

The cognitive systems in IPSEITY can be decomposed into several plugins. Each of them takes part in the decision process. As shown in Fig. 1b, 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[15].



(a) IPSEITY



(b) RL Cognitive System

**Fig. 1.** Architecture of IPSEITY and architecture of a RL cognitive system

## 2.3 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.

**Easy integration:** 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

[15] At `http://www.ipseity-project.com/docs/IpseityTechnicalGuide.pdf`.

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 [3]: remote cognitive systems with behaviors learnt using Ipseity communicate using TCP-IP sockets with agents of 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.

**Graphical interface:** Ipseity provides a user-friendly interface (see Fig.2) with informative icons and widgets for setting up all the parameters involved in the simulation of a MAS, including those of a cognitive system.

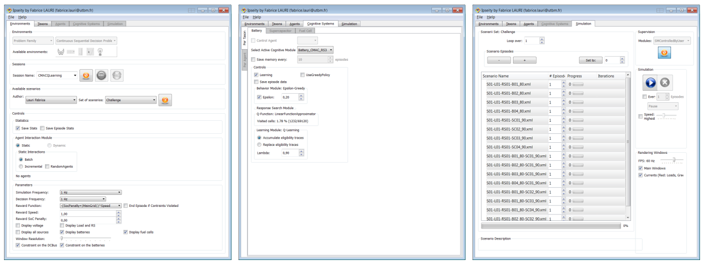


**Fig. 2.** Screenshots of some tabs in Ipseity.

## 3 Conclusion

An overview of Ipseity has been presented in this article, focusing on RL. Ipseity is highly modular and broadly extensible. 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. http://www.ipseity-project.com.
2. http://www.qt-project.org.
3. 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.
4. 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.