JBotEvolver: A Versatile Simulation Platform for Evolutionary Robotics

Miguel Duarte, Fernando Silva, Tiago Rodrigues, Sancho Moura Oliveira, and Anders Lyhne Christensen

1Instituto de Telecomunicações, Lisboa, Portugal
2Instituto Universitário de Lisboa (ISCTE-IUL), Lisboa, Portugal
3LabMAg, Faculdade de Ciências, Universidade de Lisboa, Portugal

This paper introduces JBotEvolver, a versatile simulation platform for research and education in evolutionary robotics (ER). JBotEvolver is a Java-based open-source, cross-platform framework available at https://code.google.com/p/jbotevolver/ under the GNU GPL. JBotEvolver has been used in a number of previous ER studies of our research group, from offline evolution to online evolution and learning, and from single to multirobot systems (for examples see Duarte et al. (2014a,b); Silva et al. (2012a,b)), and in a number of undergraduate and graduate courses at ISCTE-IUL.

JBotEvolver’s main features are its ease of installation and use, and its versatility in terms of customization and extension. A fundamental design philosophy behind JBotEvolver is to provide a basis for ER experiments without the need for detailed framework-specific knowledge. Following this philosophy, JBotEvolver enables the configuration of experiments programmatically or via a plaintext file that specifies which features will be included in the simulation. The corresponding classes are then seamlessly loaded in execution time via Java’s Reflection API. In this way, JBotEvolver can also make use of external, user-defined classes that extend the base implementation. Additionally, JBotEvolver is self-contained, but can also be used as an external library in other applications. One example is the automation system that allowed us to evolve hierarchical controllers by automatically combining controllers from independent evolutionary setups (Duarte et al., 2014a).

The user can extend the main components of JBotEvolver such as the environments in which the robots operate, the physical objects of the environment, the robot models, the evaluation functions, the evolutionary algorithms, and the type and structure of the controllers. For instance, while our studies have focused on the evolution of neural network-based controllers (Duarte et al., 2014b,a; Silva et al., 2012a,b), JBotEvolver does not preclude other approaches such as genetic programming or evolutionary fuzzy systems. JBotEvolver features a 2D differential-drive kinematics engine that has been used to simulate multirobot systems with up to thousands of robots (Duarte et al., 2014b). The 2D engine can also be extended or replaced by user-defined engines with different dynamics. With respect to the experimental setup, JBotEvolver allows the user to control the degree of realism of the simulations by allowing, for instance, the use of sensors modelled based on samples from real robots (Duarte et al., 2014a) and the configuration of the robots’ control cycle frequency. In terms of robots, JBotEvolver includes a model of the e-puck robot and allows for fully customizable 2D robot models.

An important characteristic of JBotEvolver is that controller evaluations are defined as tasks that can be executed sequentially or in parallel on a workstation. We have also included a connector for Conillon (Silva et al., 2011), a lightweight platform for distributed computing that enables a significantly speed-up of evolutionary processes through

Figure 1: Result Viewer GUI: (a) file tree for navigating experimental results, (b) options to load or edit files, and to plot the fitness of controllers, (c) text area for overriding experimental arguments, (d) basic 2D visual renderer, (e) options to start/stop the experiments, quit the simulator, and plot neural network activity, and (f) sliders to change the speed of the simulation and to fast-forward/rewind.
parallelization. We have used Conillon to distribute JBotEvolver tasks to over 400 cores. Conillon’s dynamic request of Java classes allows tasks with different codebases to be submitted. Worker nodes can be added to the computing network in an ad-hoc manner either through: (i) a stand-alone application, (ii) a Java applet running in a browser, or (iii) as a screensaver on PCs. In addition, the Encog framework’s implementation of the neuroevolutionary NEAT algorithm (Stanley and Miikkulainen, 2002) has been interfaced with JBotEvolver.

A number of alternative simulators for evolutionary robotics are available, including: (i) Simbad (Hugues and Bredeche, 2006), (ii) Webots (Michel, 2004), (iii) ARGoS (Pinciroli et al., 2012), (iv) Enki, and (v) the Player Project (Gerkey et al., 2003), which includes the 2D simulator Stage and the 3D simulator Gazebo. In comparison with such platforms, as described in Table 1, the key advantages of JBotEvolver are its extensibility, ease of use, and versatility. JBotEvolver has a number of expert-oriented features, such as the mechanisms for the distributed execution of experiments, and non-expert-oriented features. From the GUI (Fig. 1), it is, for instance, possible to navigate between results of different evolutionary runs or setups, analyze fitness score plots, visualize the controllers’ input and output values, stop and resume experiments, and modify experimental configurations on-the-fly.

To summarize, JBotEvolver is a versatile, easy to deploy and operate simulation platform intended for both expert and non-expert users. In our ongoing work, we continue to use JBotEvolver, and we are extending it to support different robot models and types of robots, including aquatic drones.

Acknowledgements This work was partially supported by Fundação para a Ciência e Tecnologia under the grants SFRH/BD/76438/2011, SFRH/BD/89573/2012, PEst-OE/EEI/LA0008/2013, PEst-OE/EEI/UI0434/2014, and EXPL/EEI-AUT/0329/2013.

References


<table>
<thead>
<tr>
<th>Features</th>
<th>Simulator</th>
<th>Simbad</th>
<th>Webots</th>
<th>ARGoS</th>
<th>Player</th>
<th>Enki</th>
<th>JBotEvolver</th>
</tr>
</thead>
<tbody>
<tr>
<td>2D/3D</td>
<td>3D</td>
<td>2D+3D</td>
<td>2D+3D</td>
<td>2D+3D</td>
<td>2D</td>
<td>2D</td>
<td></td>
</tr>
<tr>
<td>Open-source Dependencies</td>
<td>Java 3D</td>
<td>no</td>
<td>multiple</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>Platforms</td>
<td>win/mac/linux</td>
<td>win/mac/linux</td>
<td>multiple*</td>
<td>mac/linux</td>
<td>multiple</td>
<td>multiple*</td>
<td></td>
</tr>
<tr>
<td>Language</td>
<td>Java</td>
<td>low</td>
<td>low</td>
<td>low</td>
<td>C++</td>
<td>Java</td>
<td></td>
</tr>
<tr>
<td>Learning curve</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>high</td>
<td>low</td>
<td></td>
</tr>
<tr>
<td>Distributed evolution</td>
<td>rich</td>
<td>rich</td>
<td>medium</td>
<td>medium</td>
<td>no</td>
<td>basic</td>
<td></td>
</tr>
<tr>
<td>GUI</td>
<td>rich</td>
<td>rich</td>
<td>rich</td>
<td>rich</td>
<td>medium</td>
<td>basic</td>
<td></td>
</tr>
</tbody>
</table>

Table 1: Comparison of features between simulators. *Webots allows development in C, C++, Java, Python, Matlab, and URBI.

1Encog: http://www.heatonresearch.com/encog

2Enki: http://home.gna.org/enki/