DOI: http://dx.doi.org/10.7551/978-0-262-32621-6-ch035

JBotEvolver: A Versatile Simulation Platform for Evolutionary Robotics

Miguel Duarte[1,2], Fernando Silva[1,3], Tiago Rodrigues[1,2],
Sancho Moura Oliveira[1,2], and Anders Lyhne Christensen[1,2]
[1]Instituto de Telecomunicações, Lisboa, Portugal
[2]Instituto Universitário de Lisboa (ISCTE-IUL), Lisboa, Portugal
[3]LabMAg, Faculdade de Ciências, Universidade de Lisboa, Portugal
miguel_duarte@iscte.pt

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 multi-robot 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.

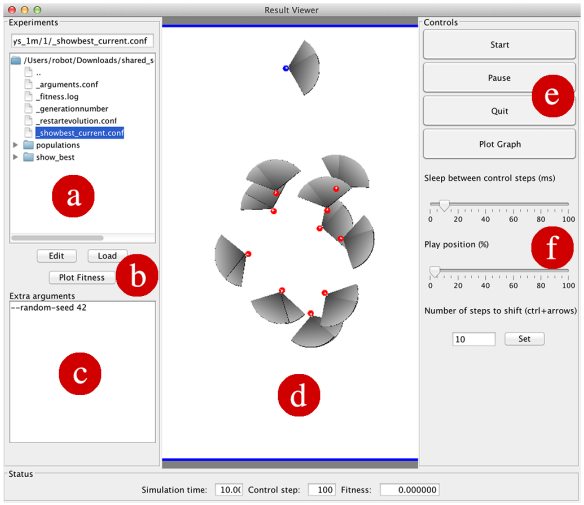

Figure 1: Result Viewer GUI: (a) file tree for navigating experimental results, (b) options to load or edit files, and to plot the fitness of controllers, (c) text area for overriding experimental arguments, (d) basic 2D visual renderer, (e) options to start/stop the experiments, quit the simulator, and plot neural network activity, and (f) sliders to change the speed of the simulation and to fast-forward/rewind.

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

Features | Simulator	Simbad	Webots	ARGoS	Player	Enki	JBotEvolver
2D/3D	3D	2D+3D	2D+3D	2D+3D	2D	2D
Open-source	yes	no	yes	yes	yes	yes
Dependencies	Java 3D	multiple	multiple	multiple	multiple	none
Platforms	win/mac/linux	win/mac/linux	mac/linux	mac/linux	mac/linux	win/mac/linux
Language	Java	multiple*	C++	C++	C++	Java
Learning curve	low	intermediate	high	high	intermediate	low
Distributed evolution	no	no	no	no	no	yes
GUI	rich	rich	medium	medium	basic	rich

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

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 standalone application, (ii) a Java applet running in a browser, or (iii) as a screensaver on PCs. In addition, the Encog framework[1] 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[2], 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

Duarte, M., Oliveira, S. M., and Christensen, A. L. (2014a). Evolution of hybrid robotic controllers for complex tasks. *Journal of Intelligent and Robotic Systems*. In press.

Duarte, M., Oliveira, S. M., and Christensen, A. L. (2014b). Hybrid control for large swarms of aquatic drones. In *Proceedings of the Fourteenth International Conference on the Synthesis & Simulation of Living Systems (ALIFE)*. MIT Press, Cambridge, MA. In press.

Gerkey, B., Vaughan, R. T., and Howard, A. (2003). The player/stage project: Tools for multi-robot and distributed sensor systems. In *Proceedings of the Eleventh International Conference on Advanced Robotics (ICAR)*, pages 317–323. FCT/UC, Coimbra, Portugal.

Hugues, L. and Bredeche, N. (2006). Simbad: an autonomous robot simulation package for education and research. In *Proceedings of the Ninth International Conference on the Simulation of Adaptive Behaviour (SAB)*, pages 831–842. Springer, Berlin, Germany.

Michel, O. (2004). Webots: Professional mobile robot simulation. *International Journal of Advanced Robotic Systems*, 1(1):39–42.

Pinciroli, C., Trianni, V., O'Grady, R., Pini, G., Brutschy, A., Brambilla, M., Mathews, N., Ferrante, E., Di Caro, G., Ducatelle, F., Birattari, M., Gambardella, L., and Dorigo, M. (2012). ARGoS: a modular, parallel, multi-engine simulator for multi-robot systems. *Swarm Intelligence*, 6(4):271–295.

Silva, F., Urbano, P., and Christensen, A. L. (2012a). Adaptation of robot behaviour through online evolution and neuromodulated learning. In *Proceedings of the Thirteenth Ibero-American Conference on Artificial Intelligence (IBERAMIA)*, pages 300–309. Springer, Berlin, Germany.

Silva, F., Urbano, P., Oliveira, S., and Christensen, A. L. (2012b). odNEAT: An algorithm for distributed online, onboard evolution of robot behaviours. In *Proceedings of the Thirteenth International Conference on the Simulation & Synthesis of Living Systems (ALIFE)*, pages 251–258. MIT Press, Cambridge, MA.

Silva, H., Oliveira, S. M., and Christensen, A. L. (2011). Conillon: A lightweight distributed computing platform for desktop grids. In *Proceedings of the Sixth Iberian Conference on Information Systems and Technologies (CISTI)*, pages 1–6. IEEE Press, Piscataway, NJ.

Stanley, K. and Miikkulainen, R. (2002). Evolving neural networks through augmenting topologies. *Evolutionary Computation*, 10(2):99–127.

[1]Encog: `http://www.heatonresearch.com/encog`

[2]Enki: `http://home.gna.org/enki/`