Resource Review: the ECJ Toolkit

David R. White

Introduction

ECJ Homepage: http://cs.gmu.edu/~eclab/projects/ecj/

ECJ is one of the most popular evolutionary computation toolkits and one that is widely used within the field of Genetic Programming (GP). In keeping with previous reviews, this article is a critical assessment of the features of ECJ, its main strengths and its shortcomings.

Background

ECJ has a long history: development began in 1998, and the last review of ECJ within GPEM was published in 2004 [4]. Since that review, eleven major versions of the toolkit have been released and many new features added. ECJ is probably the toolkit with the most complete support for GP.

ECJ is available for download as a 4 megabyte compressed archive, or from a subversion repository, and is almost entirely released under the Academic Free License. The development team is headed by Sean Luke at George Mason University. It is difficult to measure ECJ's popularity, partly because there isn't a unique published paper that is cited by its end users. However a Google Scholar search for the current ECJ URL returned 251 papers mentioning the website.

Major Features

ECJ is a generic toolkit aimed at all forms of evolutionary computation. As well as GP, it supports Genetic Algorithms, Evolutionary Strategies, Particle Swarm Optimisation and Differential Evolution. However, GP is one of its strongest suits and correspondingly ECJ supports many useful GP features such as strongly-typed GP using atomic and set typing, ephemeral random constants (ERCs) and automatically defined functions (ADFs). It is currently focused on tree-based GP, although support for grammar-based representations is improving. It does not provide direct support for Linear-GP.

Due to its well-engineered structure, which makes heavy use of Java inheritance, abstraction and pattern-oriented design, ECJ is able to support many alternative methods for common functions, such as population initialisation, selection and variation operators, without requiring additional user-written code. Multi-objective optimisation, coevolution and parsimony measure are included. Where appropriate, ECJ uses Koza's conventions and parameter settings.

Strengths

Since the last GPEM review involving ECJ, the tool has been expanded and greatly improved. It now supports multi-thread evaluation and breeding, a master-slave architecture, island models, and even experimental support for GPGPU through a third-party extension. This is typical of ECJ's advancement, which has put it at the forefront of EC tools.

The use of simple text-based parameter files make the system the most configurable I have come across. Thus many uses of ECJ require little programming. For example, multi-thread evaluation can often be introduced with the line `eval.threads=5` in a simple text file. The parameter files can also be arranged hierarchically. This helps to hide the complexity of the sheer number of configurable options but can also lead to confusion amongst new users. It can also occasionally cause problems even for experienced ECJ users when a parameter elsewhere in the hierarchy produces an unexpected result.

Careful documentation and community participation are vital to make even a well-written piece of software usable. The documentation of ECJ makes it stand out from other similar toolkits. For new users, four well-explained tutorials give a rapid introduction to ECJ's basic features, including parameter files, breeding pipelines and statistics reporting. Standard GP problems such as symbolic regression and Boolean functions are provided as illustrative examples. The owner's manual's 204 pages thoroughly document the system. They are mostly of use to people who want to understand the internals of ECJ or extend it in a generic way. The owner's manual complements the Javadoc and verbosely-commented source code. ECJ also has a very strong mailing list with nearly 300 subscribers, averaging 17 posts a month during the last year, and Sean Luke is quick to supply helpful advice. Most queries are answered within a day of posting.

This strong documentation and the general extensibility of the framework has led to many contributions from third parties, including support for the multi-objective algorithm SPEA2, variants such as Cartesian Genetic Programming (CGP) and Gene Expression Programming (GEP) and even running on graphics cards (GPGPU).

Its implementation in Java makes ECJ very portable. Unless ECJ's graphical user interface (GUI) is needed, ECJ is self contained. (The GUI requires a few extra jar files from the web and a one-line remake of the source.) The integration of networking and serialisation support that Java provides makes developing new parallel architectures and checkpointing methods much easier than starting from scratch or using a third-party library.

Weaknesses

The last review of ECJ in GPEM observed that a steep learning curve was involved in grasping the design of the whole system. ECJ is a large extensible framework and it takes time to understand its idioms, therefore, despite the improvements in documentation, it remains difficult use at first. If you are only interested in applying GP to a new problem, ECJ might be too powerful. Perhaps reflecting its userbase, the ECJ GUI is one of the most neglected parts of the system. Simple features, such as easy access to recently used parameter files, are not implemented. I personally only use the GUI when interested in displaying charts. (ECJ provides the skeleton code, using JFreechart, to show charts.) Similarly, post-analysis and visualisation are not ECJ's strong suit.

The cost of using Java in such a strongly object-oriented design is to make ECJ memory hungry. It can also be a little slow occasionally. This is probably due to the garbage collection of a large number of short-lived objects.

Despite following many object-oriented code conventions to the letter, ECJ does not use Java's polymorphic generics facility. This has proven to be a controversial decision. Whilst the efficiency issues of using generics are debatable, ECJ code is prone to a lot of casting that the evolutionary computation Watchmaker framework [2], which uses generics, does not suffer from.

Conclusion

When asked to recommend a Java toolkit for GP, I usually suggest ECJ. A novice may prefer a simpler and less powerful alternative such as GPLAB [3, **?**] or Eureqa [1]. If you are a programmer or researcher then in terms of features, extensibility, documentation and support, ECJ is unsurpassed.

References

[1] R. Dubčáková. Eureqa: Software Review. *Genetic Programming and Evolvable Machines*, 12:173–178, 2011.

[2] D. W. Dyer. http://watchmaker.uncommons.org/, 2011.

[3] S. Silva and J. Almeida. GPLAB - a Genetic Programming Toolbox for Matlab. In *Proceedings of the Nordic MATLAB Conference*, pages 273–278, 2005.

[4] G. C. Wilson, A. McIntyre, and M. I. Heywood. Resource Review: Three Open Source Systems for Evolving Programs–Lilgp, ECJ and Grammatical Evolution. *Genetic Programming and Evolvable Machines*, 5(1):103–105, 2004.