Genet Program Evolvable Mach (2014) 15:215–218
DOI 10.1007/s10710-014-9214-4

SOFTWARE REVIEW

# Software review: the HeuristicLab framework

**Achiya Elyasaf · Moshe Sipper**

Received: 15 January 2014 / Revised: 15 January 2014 / Published online: 24 January 2014


## 1 Introduction

HeuristicLab homepage: http://dev.heuristiclab.com

Computer scientists often find themselves debating whether they should use (and extend) an existing system or develop their own. The onerous task of learning how to operate an existing system can be a severe overhead when all one wishes is to conduct a small experiment. However, especially where extended experiments are to be conducted, in the long run, a system that is extensible, flexible, modular, and usable, can save valuable time.

HeuristicLab [6, 7] is a graphical user interface (GUI) based framework for heuristic and evolutionary algorithms, designed for ease of use.

It offers a plugin-based architecture (which enables users to add custom extensions without knowing the whole of the source code), a domain-independent model to represent arbitrary search algorithms, support for graphical user interfaces, and the ability to accommodate parallel algorithms.

HeuristicLab has been under development since 2002 by members of the Heuristic and Evolutionary Algorithms Laboratory (HEAL) [2], at the Upper Austria University of Applied Sciences. The team now includes six dedicated developers, who publish new versions with bug fixes and new features in a frequent, timely fashion. Additionally there are three major version releases per year. Thus, any reported bug is usually fixed almost immediately. Similarly many requested features are quickly implemented. HeuristicLab is downloaded about 70 times a week, and the number of papers citing the system is rapidly increasing. HeuristicLab

A. Elyasaf (✉) · M. Sipper
Ben-Gurion University of the Negev, Be'er Sheva, Israel
e-mail: achiya.e@gmail.com

M. Sipper
e-mail: sipper@cs.bgu.ac.il

is free, with source code, documentation, etc. being distributed under the terms of a GNU General Public License (GPL).

Below we survey the major features and advantages of HeuristicLab, as well as a number of weaknesses. In addition, we compare HeuristicLab to another popular evolutionary system, namely, ECJ [1].

## 2 Major features

The salient features of HeuristicLab [7] are:

- *Rich graphical user interface* A comfortable and feature-rich graphical user interface reduces the learning effort and enables users without programming skills to use and apply HeuristicLab.
- *Many algorithms and problems* Several well-known heuristic algorithms and optimization problems are already implemented in HeuristicLab and can be used off-the-shelf (e.g., Genetic Algorithms, GP, Tabu Search, Simulated Annealing and Hill Climbing).
- *Extensibility* HeuristicLab consists of many different plugins. Users can create and reuse plugins to integrate new features and extend the functionality of HeuristicLab.
- *Visual algorithm designer* Optimization algorithms can be modeled and extended with the graphical algorithm designer.
- *Experiment designer* Users can define and execute large experiments by selecting algorithms, parameters and problems in the (graphical) experiment designer.
- *Results analysis* HeuristicLab provides interactive charts for quickly and thoroughly analyzing results.
- *Parallel and distributed computing* HeuristicLab supports parallel execution of algorithms on multi-core and cluster systems (although currently there is no support for GPUs—Graphics Processing Units).

## 3 Strengths

HeuristicLab abounds with excellent features, the two we found to be most important are:

The simple yet feature-rich GUI is probably the first strength that new users immediately notice. While there are other systems for the task of designing heuristics and evolutionary algorithms [4], they all require extensive programming skills. With HeuristicLab one can mostly forgo programming and instead use the GUI. The GUI enables the user to: (1) define the representation (genotype) and genetic operators; (2) create experiments that examine the effectiveness of operators (e.g., comparing different types of selection methods); (3) run, pause, and resume experiments at will; (4) generate statistics and graphs, used to analyze results; and, (5) create new algorithms. Achieving all this mostly by interacting graphically

rather than by laboriously hacking code renders the system a pleasure to work with. When one needs to program new code (e.g., a fitness evaluation function) HeuristicLab offers an easy interface to do so. There is no need to edit and compile the entire source code nor is it necessary to write a class and inherit another one. It is only necessary to write the function through the GUI.

The second major strength is the HEAL team itself. As mentioned in Sect. 1 there are six developers working on this system, and they are also highly active in the HeuristicLab discussion group [3]. Thus, most of the reported bugs or questions get fixed or answered in less then a week (we can attest to this personally). We find the experience of daily builds and rapid support to be unique.

## 4 Weaknesses

HeuristicLab is written in Microsoft C# and therefore works best under the Microsoft Windows operating system using Microsoft's .NET Framework. However, Linux and Apple users have not been totally ignored, as the Mono project [5] offers an open-source implementation of the .NET Framework. Even though HeuristicLab with Mono suffers from some limitations, most of the features do work, and there are workarounds for those that do not. The HeuristicLab team is constantly working on improving the Mono version.

As we see it, the major weakness of HeuristicLab is the lack of proper documentation. There are several video tutorials, a superb discussion group, a paper with in-depth explanations [7], and some tips and documentation scattered about the project website. However, a complete API or even a cookbook that takes you slowly and methodically through all the features and tricks is crucial for novice and intermediate users. When it is necessary to modify the source code, e.g. in order to write complex genetic operators or genotypes, it may take some time to track down the right C# classes that need to be inherited. We would be delighted to see more documentation.

## 5 Comparison to other systems

Parejo et al. [4] compared several systems for heuristic and evolutionary algorithms. According to their thorough study the top two systems are HeuristicLab and ECJ. ECJ [1] is a popular Java-based evolutionary computation research system, which has been under development since 1998 by Sean Luke and his colleagues at the Evolutionary Computation Laboratory, George Mason University. ECJ has excellent documentation.

Because HeuristicLab is designed to be used mostly through a GUI it has some major advantages over ECJ, as is clearly concluded by Parejo et al. [4]. The GUI enables easy encoding of new problems and adapting of existing ones; it also enables executing algorithms in parallel and tuning experimental parameters.

HeuristicLab's lesser popularity and dearth of documentation are its two major weaknesses. A somewhat minor weakness revealed by Parejo et al. [4] is the limited

support for HeuristicLab in platforms other than Microsoft Windows. These strengths and weaknesses accord with our own experience.

HeuristicLab is very easy to install and run: just download the package and run the executable file. This is not the case with ECJ, where you need to download and extract ECJ and optional support libraries, and tune various configuration files, class paths, and build the sources. Although these steps might seem easy to experienced programmers, this is not so for novice users.

In [8], White states that "If you are only interested in applying GP to a new problem, ECJ might be overkill". This does not apply to HeuristicLab since creating new problems, tuning, and running them are accomplished with a few key strokes.

## 6 Conclusion

HeuristicLab is an emerging system that is rapidly gaining popularity. The system is surprisingly fun to use, and offers an easy way to create new evolutionary algorithms, run them, and analyze the results.

## References

1. ECLab Evolutionary Computation Laboratory, George Mason University: ECJ 2.0 (2010). http://cs.gmu.edu/eclab/projects/ecj/
2. HEAL: Heuristic and Evolutionary Algorithms Laboratory. http://heal.heuristiclab.com/
3. HEAL (Heuristic and Evolutionary Algorithms Laboratory): Official HeuristicLab discussion group. http://groups.google.com/forum/?fromgroups=#!forum/heuristiclab/
4. J.A. Parejo, A. Ruiz-Cortés, S. Lozano, P. Fernández, Metaheuristic optimization frameworks: a survey and benchmarking. Soft Comput. **16**(3), 527–561 (2011). doi:10.1007/s00500-011-0754-8
5. The Mono project website. http://www.mono-project.com
6. S. Wagner, Heuristic optimization software systems - modeling of heuristic optimization algorithms in the HeuristicLab software environment. Ph.D. thesis, Johannes Kepler University, Linz, Austria (2009)
7. S. Wagner, G. Kronberger, A. Beham, M. Kommenda, A. Scheibenpflug, E. Pitzer, S. Vonolfen, M. Kofler, S. Winkler, V. Dorfer, M. Affenzeller, in Architecture and design of the HeuristicLab optimization environment, ed. by R. Klempous, J. Nikodem, W. Jacak, Z. Chaczko. Advanced Methods and Applications in Computational Intelligence, Topics in Intelligent Engineering and Informatics, vol 6. (Springer, New York, 2014), pp. 197–261. doi:10.1007/978-3-319-01436-4_10.
8. D.R. White, Software review: the ECJ toolkit. Genet. Program. Evolvable Mach. **13**(1), 65–67 (2012). doi:10.1007/s10710-011-9148-z