

Evolutionary Computation

Stefano Benedettini

Andrea Roli

Artificial Intelligence LM 2010

Contents

1	Python Tools for Scientists [optional]	1
2	Python Tools for Evolutionary Computation	2
3	Evolutionary Computation Challenges	2

Topics

- Genetic Algorithm
- Genetic Programming
- Application of Evolutionary Computation
- Python tools for EC

1 Python Tools for Scientists [optional]

A number of scientific tools are available for the `Python` platform that will turn your “favourite” dynamic language into a viable, free, open-source Matlab, only much more powerful and expressive.

These libraries are completely optional and not required to advance in the assignment. Nevertheless they should earn a place in your toolbox if you are remotely into science.

Numpy: `numpy` is a library that delivers an efficient implementation of multidimensional arrays combined with a natural and convenient `Python` syntax. Many mathematical and statistical functions are also provided.

Matplotlib: this library, dependant on `numpy`, allows you to produce high quality graphics. Warmly recommended.

SciPy: SciPy is a mathematical library that incorporates `numpy` and `matplotlib` in order to offer a broad scope environment that accommodates for many different needs like: linear algebra, interpolation, statistics, etc. . . SciPy is also home of many side projects that can improve your user experience and productivity like the IPython interactive shell.

2 Python Tools for Evolutionary Computation

One problem with Evolutionary Computation is that algorithms and techniques are tricky and tedious to implement. On the contrary, one of the major advantages is that EC methods are well studied and understood and, differently from other optimization techniques, like local search for instance, software framework *do* exist. In these exercises we will focus on one of such frameworks, namely Pyevolve. **Pyevolve** is fairly easy to use and supports either genetic algorithms and (tree-based untyped) basic genetic programming.

In the link above you can find documentation, installation instructions and a tutorial. This blog, by the same author of **Pyevolve**, contains plenty of material.

For Genetic Programming, also this Python library is available: <http://pystep.sourceforge.net/>.

3 Evolutionary Computation Challenges

This session is organized as a collection of “challenges” in the sense that they are not exercises *per se* but opportunities to dig into EC topics by doing stuff. Therefore, you are encouraged to try the examples in the **Pyevolve** tutorial, modify them, see what happens and make sense out of them. For example, after you have acknowledged that your GA algorithm performs well for on a certain problem, try to modify the problem in order to make it difficult for your implementation. After that, you can try to tweak your algorithm in order to adjust it to the new problem.

Moreover, these challenges can give you hints about the possibilities of EC.

Install Pyevolve: Yes, this is a challenge. A rather dull one, but necessary nonetheless.

Learn to use Pyevolve: Read the tutorial and experiment with the examples.

EC for Combinatorial Optimization: Genetic algorithms have been successfully applied to several hard combinatorial problems. To begin with, you can use a Holland’s simple GA to maximize a function of the genotype. First try a very simple function like “counting the ones”, then increase the complexity. Remember the lesson about epistasis and engineer a function in which the dependency between genes is strong.

Variable Fixing and Master-slave Algorithm Framework: The general idea of Master-slave framework is that it is possible to split the solution construction in two nested phases. In the first phase, the parameters of a solution construction procedure are

set by a *master* solver and in the second phase the solution is actually built by a *slave* solver. The master algorithm can be a GA while the slave can implement actual evaluation of each genotype produced by the GA. Variable fixing, as its name suggests, involves in assigning some decision variables and solving the problem, usually to optimality, on the remaining variables.

EC for Continuous Optimization: By choosing appropriate genotype representation and operators, continuous functions can be optimized too, from simple two dimensional convex to hard “bumpy” fifty-dimensional functions like those making up the test suite for CEC 2008 Special Session on Global Optimization.

EC for parameter Tuning: partially related to Master-slave algorithms, EC techniques can be used to tune parameters of an algorithm.

EC for Automatic Generation of Hard Problem Instances: Given appropriate definitions, GA can “evolve” pretty much everything. In this case, you aim to find harder and harder problem instances for a given algorithm. Fitness function should then measure the “difficulty” of such instances for the chosen algorithm.

Genetic Programming for Symbolic Interpolation: Use genetic programming to interpolate a set of points with a composition of simpler functions.

Genetic Programming for Automatic Synthesis of Controllers: Partially related to the previous topic, you can use GP to evolve more than simple functions. With GP you can try to evolve a controller for a system (not unlikely to what you have done in control courses). The evaluation, in this case, is a “simulation” of the whole controller+system.