A Graphical User Interface for Evolutionary Algorithms

Zoltán Tóth

Engineering applications provide a wide range of optimization problems for people working in this area. In most cases, the different tasks require different programming environments to achieve the best results.

The purpose of *Generic Evolutionary Algorithms Programming Library (GEA)* system is to provide researchers with an easy-to-use, widely applicable and extendable programming library which solves these tasks by means of evolutionary algorithms.

Evolutionary algorithms (EAs) are general purpose function optimization methods which search for optima by making potential solutions (individuals) compete for survival in a population. The better an individual is, the better chance it has to survive. The search space is explored by modifying the potential solutions by genetic operators observed in nature: generally mutation and recombination.

Evolutionary algorithms have (among others) the following two advantages over other optimization methods: first, in most cases they converge to global optima, and second, the usage of the black-box principle (which only requires knowledge about a function's input and output to perform optimisation on it) makes them easily applicable to functions whose behavior is too complex to handle with other methods.

The *GEA* system contains algorithms for various evolutionary methods, implemented genetic operators for the most common representation forms for individuals, various selection methods, and examples on how to use and expand the library. An extensive comparison of *GEA* and other evolutionary algorithm implementations shows that *GEA* can be effectively applied on many optimization problems.

The implemented genetic operators, selection methods and evolutionary algorithms make the system easy-to-use even for beginners: If the user wants to solve a problem with *GEA* and the search space consists of, say, bit-strings or real vectors, then he/she only has to define the problem-specific fitness function, set the parameters of the algorithm and start searching for the solution.

GEA is implemented in the `ANSI C++` programming language. The class hierarchy and the applied plug-in technology make the extension of the system with new selection methods, representation forms for individuals or even evolutionary algorithms as easy as possible.

GraphGEA is a graphical user interface to *GEA* written with the GTK API. The numerous parameters of the evolutionary algorithm can be set in appropriate dialog boxes. The program also checks the correctness of the parameters and saving/restoring of parameter sets is also possible. The selected evolutionary algorithm can be executed interactively on the specified optimization problem through the graphical user interface of *GraphGEA*, that is, the execution can be controlled by appropriate buttons and the results and behavior of the EA can be observed on several selected graphs (on-line visualization). The available graphs include several fitness and distribution graphs as well as depictions of the genotypes and phenotypes of individuals.

GraphGEA is capable of off-line visualization, too, i.e. it gathers and saves information during the run of the evolution process and can reconstruct the run from this information later.

It is very important to mention that *GraphGEA* is "only" a graphical user interface to *GEA* and is not necessary for the using of *GEA*. *GEA* runs as a child process of *GraphGEA* and the processes communicate by means of a pipe system and shared memory (UNIX System V IPC).

While the main purpose of *GEA* is solving optimization problems, that of *GraphGEA* is education and analysis. It can be great help for students understanding the characteristics of evolutionary algorithms and researchers of the area can use it to analyze an EA's behavior on particular problems.