Chapter XL
An Object–Oriented Framework for Rapid Genetic Algorithm Development

Andrés L. Medaglia
Universidad de los Andes, Colombia

Eliécer Gutiérrez
Universidad de los Andes, Colombia

ABSTRACT

JGA, the acronym for Java Genetic Algorithm, is a computational object-oriented framework for rapid development of evolutionary algorithms for solving complex optimization problems. This chapter describes the JGA framework and illustrates its use on the dynamic inventory lot-sizing problem. Using this problem as benchmark, JGA is compared against three other tools, namely, GAlib, an open C++ implementation; GADS, a commercial Matlab® toolbox; and PROC GA, a commercial (yet experimental) SAS® procedure. JGA has proved to be a flexible and extensible object-oriented framework for the fast development of single (and multi-objective) genetic algorithms by providing a collection of ready-to-use modules (Java classes) that comprise the nucleus of any genetic algorithm. Furthermore, JGA has also been designed to be embedded in larger applications that solve complex business problems.

INTRODUCTION

Since the conception of genetic algorithms (GAs), researchers and practitioners alike faced the problem of building tools which could make the implementation of their own applications easier. At the beginning, the most widely used guide was the "Simple GA code" (SGA) implementation from Goldberg (1989) built in the Pascal programming language. Today, there is a broad array of offerings of genetic algorithm libraries available in different languages and computing platforms. For a thorough survey on the subject, the reader is referred to Pain and Reeves (2002).

Some of the earliest tools were coded in the C language. C evolved into C++, adopting the object-oriented programming (OOP) paradigm.

Copyright © 2007, Idea Group Inc., distributing in print or electronic forms without written permission of IGI is prohibited.

The advantage of using the C or C++ language is mainly its computer efficiency and outstanding performance in terms of speed. In this class, we found tools such as ECGA (Lobo & Harik, 1999), GALOPPS (Goodman, 2002), and GALib (Wall, 2005).

ECGA (Extended Compact GA) is a tool implemented in C++, based on the GA code from Goldberg, in which the user can replace or modify the core classes. This tool does not include templates or heritage concepts, but the class structure provides independency to every basic component of a genetic algorithm.

GALOPPS is a flexible generic GA implementation in C. To make it easier for users to learn and extend, it was also based upon Goldberg's SGA architecture. GALOPPS includes most of the chromosomal structures and operators described by Goldberg (1989). The latest version includes a parallel architecture mode in which each process can handle one or several interacting subpopulations.

GAlib contains a set of C++ classes for building genetic algorithms. GAlib provides a set of built-in representations and operators that could be extended and customized. The built-in chromosomes include binary, integer, and real arrays of variable length. It also includes more complex data structures such as lists and trees. Chromosome initialization, mutation, crossover, and comparison methods can be customized by deriving classes for the specific problem on hand. For simple applications, the user only needs to override the fitness function class. Some details need to be taken into account depending on the development platform.

Other genetic algorithm tools have been developed for commercial vendors such as Matlab and SAS. Special-purpose libraries in Matlab are known as toolboxes. These toolboxes define in Matlab's m-files the genetic components such as fitness functions, selection operators, and crossover and mutation operators. In the m-files, functions are defined taking advantage of Matlab's powerful language and basic types such as vectors and matrices. The main advantage of Matlab-based tools is the integration with the development environment, and the numerical and graphical power provided by the Matlab core engine and enhancing toolboxes. The commercial tools have a more complete set of built-in options, including multiple populations, migration and reinsertion operators, and multi-objective ranking of objective values. Some of the commercial Matlab toolboxes available are GEATbx (Pohlheim, 2005) and GADS (Mathworks, 2005). On the other hand, there are also publicly available toolboxes such as GPLAB (Silva, 2005), GAOT (Houck, Joines, & Kay, 1995), and GATbx (Chipperfield & Fleming, 1995). Another commercial vendor, the SAS Institute, released PROC GA, an experimental procedure integrated to the SAS system (SAS Institute, 2003).

Genetic algorithm tools are also available for Microsoft Excel. Some examples are the commercial tools GeneHunter (Ward Systems Group, 2003) and Evolver (Palisade Corp., 2005). The model and the objective function are specified by referencing specific cells in the worksheet. Both tools have options to redefine some components by means of Visual Basic programming or dynamic linked libraries (DLLs). These tools can also be accessed from other programs through their companion DLLs.

The Java Programming Language has had a fast penetration in the software and hardware market over the last decade. Some Java-based tools for genetic algorithms have been implemented. These packages follow many of the principles of the previously mentioned tools for C++. Two of these implementations are GGAT (Derderian, 2002) and JGAP (Rotstan & Meffert, 2005).

GGAT is a tool developed in Java with the philosophy of providing an open specification that allows users to create new components

15 more pages are available in the full version of this document, which may be purchased using the "Add to Cart" button on the publisher's webpage:
www.igi-global.com/chapter/object-oriented-framework-rapid-genetic/21155

Related Content

Schematic Classification Model of Green Computing Approaches
Nishtha Kesswaniand Shelendra Kumar Jain (2017). *Nature-Inspired Computing: Concepts, Methodologies, Tools, and Applications (pp. 1643-1650).*
www.irma-international.org/chapter/schematic-classification-model-of-green-computing-approaches/161087

AGE-P: An Evolutionary Platform for the Self-Organization of Smart-Appliance Ensembles
Ralf Salomonand Stefan Goldmann (2010). *Nature-Inspired Informatics for Intelligent Applications and Knowledge Discovery: Implications in Business, Science, and Engineering (pp. 182-203).*
www.irma-international.org/chapter/age-evolutionary-platform-self-organization/36316

Visualizing Neuroscience Through AI: A Systematic Review
Roohi Sille, Akshita Kapoor, Tanupriya Choudhury, Hussain Falih Mahdiand Madhu Khurana (2023). *Exploring Future Opportunities of Brain-Inspired Artificial Intelligence (pp. 15-27).*
www.irma-international.org/chapter/visualizing-neuroscience-through-ai/320607

Prediction of International Stock Markets Based on Hybrid Intelligent Systems
Salim Lahmiri (2017). *Nature-Inspired Computing: Concepts, Methodologies, Tools, and Applications (pp. 1651-1667).*
www.irma-international.org/chapter/prediction-of-international-stock-markets-based-on-hybrid-intelligent-systems/161088

The Worst-Case Stabilization Time of a Self-Stabilizing Algorithm under the Weakly Fair Daemon Model
Tetz C. Huang, Ji-Cherng Lin, Chih-Yuan Chenand Cheng-Pin Wang (2010). *International Journal of Artificial Life Research (pp. 45-52).*
www.irma-international.org/article/worst-case-stabilization-time-self/46028