GP-COM: A Distributed, Component-Based Genetic Programming System in C++

Christopher Harris and Bernard Buxton

Department of Computer Science
University College, London
Gower Street
London
WC1E 6BT
United Kingdom
Email: C.Harris@cs.ucl.ac.uk, B.Buxton@cs.ucl.ac.uk

ABSTRACT

The widespread adoption of genetic programming (GP) as a problem-solving method, depends on the availability of good tools and support for applying GP to existing problems. A system is presented that mirrors the conceptual structure of the genetic programming method, maximising flexibility and re-use of code. This reduces the effort required to apply GP to a new problem domain.

Structure of a GP system

Genetic programming involves a cyclical process by which populations of candidate programs are manipulated based on their ability to solve a given problem. The GP cycle can conceptually be broken down into problem-dependent and problem-independent parts (see Figure 1.).

GP-COM is a genetic programming system based around this conceptual structure. Components in the system are loosely coupled and are defined only by their interfaces to other components. Components may be easily swapped to test potential improvements to the process, or to apply the system to a enw problem domain. The system consists of a collection of components glued together by high-level scripts.

The central component is a population manager which performs all operations essential to the manipulation and management of populations of genetic programs. This component does not implement the cycle but is an 'engine' upon which are layered control scripts. The engine is flexible, and supports strongly typed primitives [Montana 1995], Automatically Defined Functions [Koza 1992] (also strongly typed), and demes. The execution of candidate programs is entirely independent of the software used to create and manipulate them. The system generates C code for each candidate, which is linked into a problem-specific program skeleton. This allows the system to be used with existing libraries of software as primitives.

GP-COM supports multiple concurrent runs through high-level UNIX processes. It can control multiple machines, of varying architecture, executing any number of simultaneous runs. A simple and flexible configuration process allows experimental parameters to be easily tuned for particular problems and runs.

The system has been successfully applied to symbolic regression problems and to edge detection in 1-dimensional signals [Harris 1996].

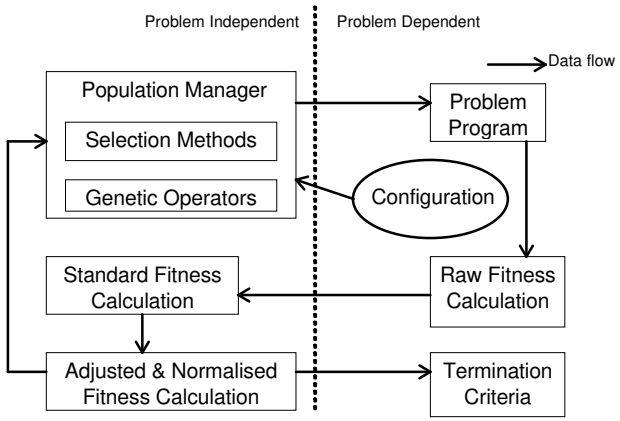

Figure 1. Conceptual structure of the GP cycle

Bibliography

Harris, C.P, Buxton, B.F. 1996. *Evolving Edge Detectors.* Department of Computer Science Research Note RN/96/3, University College London.

Koza, J.R. 1992. *Genetic Programming: On the Programming of Computers by Natural Selection.* MIT Press.

Montana, D.J. 1995. Strongly Typed Genetic Programming. *Evolutionary Computation* 3(2).