Language theory with application: Cartesian genetic programming

author: Michaela Šikulová

Genetic programming is the evolution-based machine learning method, which automaticaly generates whole programs in the given programming language. In its original form Genetic Programming (GP) has evolved programs in the form of LISP parse trees. Usually, large populations are used and crossover is used as the primary method of developing new candidate solutions from older programs. Genetic programming is used to evolve the genotype[1], for example in a symbolic regression problem (SR). SR is the problem of identifying the mathematic description of a hidden system from experimental data and is closely related to general machine learning.

Cartesian Genetic Programming (CGP) is an increasingly popular and efficient form of Genetic Programming that was developed by Julian Miller in 1999 and 2000. In its classic form, it uses a very simple integer based genetic representation of a program in the form of a directed graph. Graphs are very useful program representations and can be applied to many domains (electronic circuits, neural networks). In a number of studies, CGP has been shown to be comparatively efficient to other GP techniques. It is also very simple to implement.

CGP originaly represents programs or circuits as a two dimensional grid of program primitives. This is loosely inspired by the architecture of digital circuits called FPGAs (field programmable gate arrays). The genotype is a list of integers (and possibly parameters) that represent the program primitives and how they are connected together. CGP represents programs as graphs in which there are noncoding genes (nodes or edges). The genes are addresses in data (connection genes), addresses in a look up table of functions or additional parameters. This representation is very simple, flexible and convenient for many problems. The phenotype is candidate solution with only coding genes. When we decode a CGP genotype, many nodes and their genes can be ignored because they are not referenced in the path from inputs to outputs. These genes can be altered and make no difference to the phenotype, they are non-coding. Clearly there is a many-to-one genotype to phenotype map.

In my Ph.D. thesis, I deal with symbolic regression using catresian genetic programming and advanced evolution-based techniques like coevolution. This work will deal with formal definition of cartesian program as 9-tuple, CGP, and with properties of genotype and phenotype (and genotype-phenotype mapping) in connection with the symbolic regression problem. This work will also describe three or four ways to parse cartesian program to get solution.

1 genotype – coded version of candidate solution