Information Sciences 178 (2008) 4434–4452

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Routine high-return human-competitive automated problem-solving by means of genetic programming

John R. Koza [a,*], Matthew J. Streeter [b], Martin A. Keane [c]

[a] Stanford University, Post Office Box K, Los Altos, CA 94023, United States
[b] Genetic Programming Inc., 990 Villa Street, Mountain View, California 94041, United States
[c] Econometrics Inc., 1300 North Lake Shore No. 22B, Chicago, Illionois, United States

ARTICLE INFO

Keywords:
Genetic programming
Evolutionary computation
Automated design
Automated invention
Patented inventions
Controllers
Analog circuits

ABSTRACT

Genetic programming is a systematic method for getting computers to automatically solve problems. Genetic programming starts from a high-level statement of what needs to be done and automatically creates a computer program to solve the problem by means of a simulated evolutionary process. The paper demonstrates that genetic programming (1) now routinely delivers high-return human-competitive machine intelligence; (2) is an automated invention machine; (3) can automatically create a general solution to a problem in the form of a parameterized topology and (4) has delivered a progression of qualitatively more substantial results in synchrony with five approximately order-of-magnitude increases in the expenditure of computer time. These points are illustrated by a group of recent results involving the automatic synthesis of the topology and sizing of analog electrical circuits, the automatic synthesis of placement and routing of circuits, and the automatic synthesis of controllers as well as references to work involving the automatic synthesis of antennas, networks of chemical reactions (metabolic pathways), genetic networks, mathematical algorithms, and protein classifiers.

1. Introduction

One of the central challenges of computer science is to get a computer to solve a problem without explicitly programming it to do so. Paraphrasing Arthur Samuel—founder of the field of machine learning—this challenge [20] concerns:

How can computers be made to do what needs to be done, without being told exactly how to do it?

In his 1983 talk entitled "AI: Where It Has Been and Where It Is Going," Samuel [21] provided a criterion for success in achieving the goal by saying:

"[T]he aim [is]. . .to get machines to exhibit behavior, which if done by humans, would be assumed to involve the use of intelligence."

Genetic programming starts from a high-level statement of what needs to be done and automatically creates a computer program to solve the problem. Genetic programming uses the Darwinian principle of natural selection and analogs of recombination (crossover), mutation, gene duplication, gene deletion, and certain mechanisms of developmental biology to progressively breed an improved population over a series of many generations.

* Corresponding author. Tel.: +1 650 941 0336; fax: +1 650 941 9430.
E-mail addresses: koza@stanford.edu (J.R. Koza), matt@genetic-programming.com (M.J. Streeter), mak@sportsmrkt.com (M.A. Keane).

0020-0255/$ - see front matter © 2008 Published by Elsevier Inc.
doi:10.1016/j.ins.2008.07.028

This paper makes four points:

(1) Genetic programming (described briefly in Section 2) now routinely delivers high-return human-competitive machine intelligence (Section 3).
(2) Genetic programming is an automated invention machine (Section 4).
(3) Genetic programming can automatically create a general solution to a problem in the form of a parameterized topology (Section 5).
(4) Genetic programming has delivered a progression of qualitatively more substantial results in synchrony with five approximately order-of-magnitude increases in the expenditure of computer time (Section 6).

These points are illustrated by a group of recent results involving the automatic synthesis of:

- both the topology and sizing of analog electrical circuits (Section 7),
- placement and routing (i.e., layout) of circuits (performed automatically and simultaneously with the synthesis of the circuit's topology and sizing) (Section 8),
- automatic synthesis of parameterized topologies for a general-purpose controller (Section 9),
- automatic synthesis of parameterized topologies containing conditional developmental operators (Section 10), and
- automatic synthesis of antennas, mathematical algorithms, classifiers of protein sequences, networks of chemical reactions (metabolic pathways), and genetic networks (Section 11).

2. Background on genetic programming

Genetic programming starts with a high-level description of "what needs to be done" and automatically executes an iterative procedure in an attempt to create a computer program that does what is required [9–16].

The *preparatory steps* for a run of genetic programming are the problem-specific and domain-specific steps that are performed by the human user prior to launching a run. The *executional steps* are the problem-independent and domain-independent steps that are automatically executed during the run.

2.1. Preparatory steps

Prior to launching a run of genetic programming, the human user communicates the nature of the to-be-solved problem to the genetic programming system by means of preparatory steps.

The five major preparatory steps for genetic programming entail determining:

(1) the set of terminals (e.g., the independent variables of the problem, zero-argument functions, and random constants) available to each branch of the to-be-evolved computer program,
(2) the set of primitive functions available to each branch of the to-be-evolved computer program,
(3) the fitness measure (for explicitly or implicitly measuring the fitness of individuals in the population),
(4) certain parameters for controlling the run, and
(5) a termination criterion and method for designating the result of the run.

2.2. Executional steps

Genetic programming starts with thousands of randomly created computer programs and uses the Darwinian principle of natural selection and analogs of recombination (crossover), mutation, gene duplication, gene deletion, and certain mechanisms of developmental biology to iteratively breed an improving population.

Genetic programming breeds computer programs to solve problems by executing the following three steps:

(1) Generate an initial set (called the *population*) of compositions (typically random) of functions and terminals appropriate to the problem.
(2) Iteratively perform the following substeps (a *generation*) on the population of programs until the termination criterion has been satisfied:
    (A) Execute each program in the population and assign it a fitness value using the problem's fitness measure.
    (B) Create a new population (the next generation) of programs by applying the following operations (called *genetic operations*) to program(s) selected from the population with a probability based on fitness (with reselection allowed):
        (i) *Reproduction*: Copy the selected program to the new population.
        (ii) *Crossover*: Create a new offspring program for the new population by recombining randomly chosen parts of two selected programs.
        (iii) *Mutation*: Create one new offspring program for the new population by randomly mutating a randomly chosen part of the selected program.