Engineering Applications of Artificial Intelligence 24 (2011) 526–533

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

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

Using weighted genetic programming to program squat wall strengths and tune associated formulas

Hsing-Chih Tsai *

Department of Construction Engineering, National Taiwan University of Science and Technology, #43, Section 4, Keelung Road, R.O.C. 106, Ecological and Hazard Mitigation Engineering Researching Center Taiwan, Taipei, Taiwan

ARTICLE INFO

Article history:
Received 10 April 2010
Received in revised form
14 June 2010
Accepted 26 August 2010
Available online 21 September 2010

Keywords:
Weighted formulas
Prediction
Squat wall strength
Genetic programming
Weighted genetic programming

ABSTRACT

This study developed a weighted genetic programming (WGP) approach to study the squat wall strength. The proposed WGP evolves on genetic programming (GP), an evolutionary algorithm-based methodology that employs a binary tree topology and optimized functional operators. Weight coefficients were introduced to each GP linkage in the tree in order to create a new weighted genetic programming (WGP) approach. The proposed WGP offers two distinct advantages, including: (1) a balance of influences is struck between the two front input branches and (2) weights are incorporated throughout generated formulas. Resulting formulas contain a certain quantity of optimized functions and weights. Genetic algorithms are employed to accomplish WGP optimization of function selection and proper weighting tasks. Case studies herein focused on a reference study of squat wall strength. Results demonstrated that the proposed WGP provides accurate results and formula outputs. This paper further utilized WGP to tune referenced formulas, which yielded a final formula that combined the positive attributes of both WGP and analytical models.

1. Introduction

Soft computing approaches involve neural networks, fuzzy logic, support vector machines, genetic algorithms (GA), genetic programming (GP) and so forth. Each offers distinct merits when employed in particular application categories. Back-propagation neural network is the most familiar soft computing approach for inference tasks, from which many neural network derivatives have been developed and applied in various categories (Tsai, 2010; Mehrjoo et al., 2008; Behzad et al., 2009). However, NN has been argued primarily as a "black box" model, due to the massive number of nodes and connections within its structure. Since first proposed by Koza (1992), GP has earned significant attention in terms of its ability to model nonlinear relationships for input–output mappings. Baykasoglu et al. (2008) attempted to compare a promising set of genetic programming techniques, including Multi Expression Programming (MEP), Gene Expression Programming (GEP), and Linear Genetic Programming (LGP) (Oltean et al., 2002; Ferreira, 2001; Bhattacharya et al., 2001). Results revealed an LGP to be the most efficient algorithm among those three for the studied limestone strengths. Differences between the algorithms focus primarily on the methodology used to generate an individual. Chromosome representation, tree topology, and a linear string are used, respectively, by MEP, GEP, and LGP. However, this paper focuses on evolving programming formula types with a fully weighted connection. Although, some formulas programmed by MEP, GEP, and LGP have coefficients, such are all fixed constants (Baykasoglu et al., 2008). Several researches employed GP derivatives for problems in the construction industry. Baykasoglu et al. (2009) employed GEP for concrete strength, cost, and slump. Yeh and Lien (2009) proposed a genetic operation tree (GOT) to study concrete strength. GOT employs tree topology (as does GEP) and uses optimized coefficients different to other GP derivatives. Coefficients do not frequently/completely occur in formulas programmed by any of the aforementioned GP derivatives.

GP creates solutions as programs (formulas) to solve problems using expression/operation trees. However, coefficient constants are critical in order to balance nodal input influences in a programmed formula. This paper introduces weight coefficients to tree connections and generates a fully weighted formula. The proposed model is a weighted genetic programming (WGP) and optimized with GA.

Structural walls provide both earthquake resistance and cost advantages, and reinforced concrete squat walls are widely applied (especially in low-rise buildings) to provide shear action seismic resistance. Hsu and Mo (1985) proposed a softened truss model to predict squat wall shear strength, and Hwang et al. (2001) presented their softened strut-and-tie model as a more effective approach to squat wall modeling. Tsai followed previous research to predict squat wall strengths, using both a high-order back-propagation network (Tsai, 2009) and high-order neural network derivatives (Tsai, 2010). Generally, analytical and neural

* Tel.: +886 2 27301277/+886 2 27376663; fax: +886 2 27301074.
E-mail address: tsaihsingchih@gmail.com

0952-1976/$ - see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.engappai.2010.08.010

models present specific advantages over each other. However, analytical models must use certain assumptions in formulas to work through problems, resulting in a level of accuracy often inferior to that offered by neural network models, assuming the latter use of sufficient historical data sets. A key drawback of neural network models, however, is that predictions are invariably supplied without accompanying formulas.

This paper utilizes genetic programming, a branch of soft-computing, to achieve both prediction results and programmed formulas. Genetic programming is improved by introducing the concept of fully weighted connections that create weighted genetic programming (WGP). Another contribution of this paper is its integration of analytical and WGP formulas together to obtain prediction results/formulas. Such confers the attributes of analytical models on WGP results, i.e., tuning analytical formulas with WGP.

The remaining sections of this paper include Section 2: Proposed WGP and GA optimization; Section 3: Programming shear strengths of squat walls to study WGP capacities and further tune analytical formulas; and Section 4: Conclusions.

2. Weighted genetic programming optimized using a genetic algorithm

The WGP adopted a layer number (NL) setting (see Fig. 1). Each node x_i^1 in the first layer represented one of the input parameters (including unit parameter "1")

$$x_i^1 = \text{one}\,(1\ P_1\ P_2 \ldots P_j \ldots P_{NI}), \quad j = 0 \sim NI \tag{1}$$

where x_i^1 represents nodes in the first layer and i denotes a related node number; P_j is the jth input parameter; and NI represents the number of inputs. Each $_i^1$ node selects one attached P_j.

Layers from the second to the 'eventual' (i.e., the layer immediately following the NLth) use operator nodes to calculate values in the top–down order (see Fig. 2). For two adjacent layers, x represents front nodes, which are treated as layer inputs, and y denotes back nodes, which are treated as layer outputs. A y is calculated by operators of two front x values. Operators involve two weights (w) and one function (F). Elements in GP and WGP are different in terms of connection weight. Number of scenarios for a GP element depends on the number of function candidates and is, therefore, finite. However, WGP introduces two complex weights to balance nodal input influences. Number of scenarios for each WGP element is infinite. Such an improvement offers distinct advantages that include: (1) ability to search a wide territory range with an infinite number of combination variations and (2) the presence of weight coefficients throughout output formulas. While calculating both appropriate weights and functions for a WGP is more time consuming than for GP, such is a worthwhile effort.

The layer after the NLth is called the "eventual layer" in the final output/formula. The node in the eventual layer is either an output node (O) or an operator node (y). Therefore, parameter selections should be applied to the 2^{NL} nodes in the first layer. There are 2^{NL-1}, 2^{NL-2}, …, and 2^0 operator nodes in the 2nd, 3rd, … and eventual layers, respectively. Every operator node y is operated by a set of defined functions in order to connect to two front nodal inputs x_i and x_j with weights w_i and w_j.

$$y = F(w_i, w_j, x_i, x_j) = \text{one of} \begin{cases} f_1 = w_i x_i \\ f_2 = (w_i x_i) + (w_j x_j) \\ f_3 = (w_i x_i)(w_j x_j) \\ f_4 = (w_i x_i)/(w_j x_j) \\ f_5 = |w_i x_i|^{w_j x_j} \\ f_6 = \sin(w_i x_i) \\ f_7 = \cos(w_i x_i) \\ f_8 = \exp(w_i x_i) \\ f_9 = \log|w_i x_i| \\ \cdots \quad .. \quad \cdots \\ f_{NF} = \dfrac{1}{\sin(w_i x_i) + \cos(w_j x_j)} \end{cases} \tag{2}$$

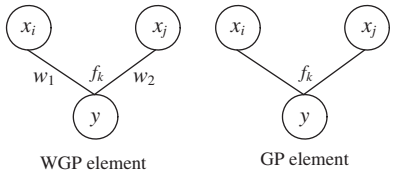

Fig. 2. Comparison for weighted genetic programming and genetic programming elements.

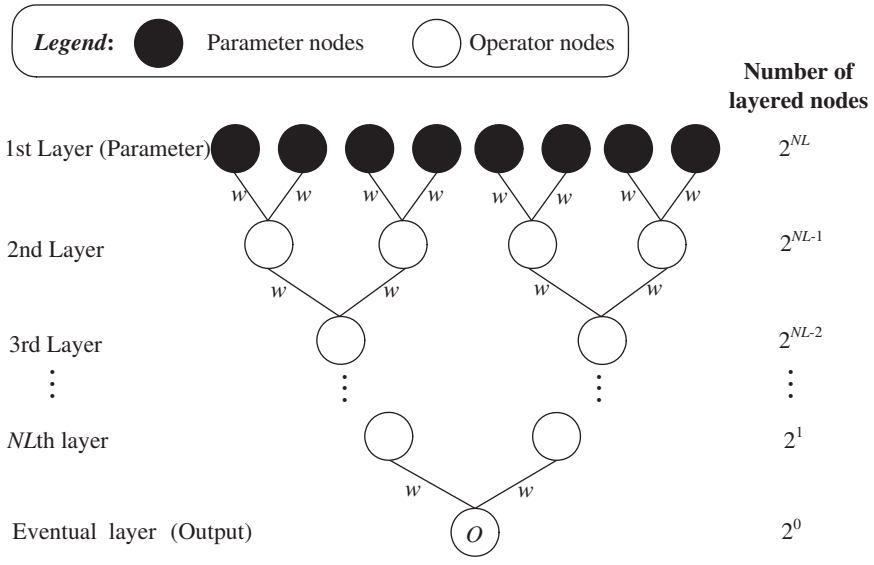

Fig. 1. Model structure for weighted genetic programming.