MODELING DEEP BEAM STRENGTHS WITH A GENETIC PROGRAMMING SYSTEM

Hsing-Chih Tsai* and Yong-Huang Lin

Department of Construction Engineering, National Taiwan University of Science and Technology, Taipei, Taiwan
** Corresponding author (tsaihsingchih@gmail.com)*

ABSTRACT: This paper proposes a genetic programming system (GPS) to both predict and program deep beam strength. The soft prediction is to provide estimated results for problems and the soft programming is to represent problems with meaningful formulas. In proposed GPS, two models of genetic programming (GP) and weighted genetic programming (WGP) are involved. GP is structured of a tree topology, inputs, and operators. WGP improves GP on introducing weights to balance tree blanches. Both GP and WGP results are provided in the GPS. Besides, this paper further designed a WGP to provide polynomial-like formulas, which seemed pretty helpful to parameter studies. Deep beam strengths were used as the case study. Results reveal that GPS has two distinct advantages over other black-box approaches in programming formulas and parameter studies. Furthermore, GPS does not surrender in prediction accuracy at all. Summarily, uses of proposed GPS are to predict, program, and model engineering problems.

***Keywords**: Genetic Programming, Weighted Genetic Programming, Genetic Programming System, Deep Beam Strengths*

1. INTRODUCTION

American Concrete Institute (ACI) code 318-95 (clause 10.7.1) [1] classifies a beam as a deep beam for flexure if the clear-span / overall-depth ratio is <1.25 for simply supported beams and 2.5 for continuous beams, and as deep beams for shear if the clear-span / effective-depth ratio is <5 for simply supported beams loaded on one face and supported on the opposite face so that compression struts can develop between loads and supports.

Neural networks (NN) are the most familiar soft computing approach for inference tasks, from which many neural network derivatives have been developed and applied in various categories [4,8,10]. However, NN has been primarily argued as a “black box” model due to the massive number of nodes and connections within its structure. Since first proposed by Koza [7], genetic programming (GP) has earned significant attention in terms of its ability to model nonlinear relationships for input-output mappings. Baykasoglu et al. [2] attempted to compare a promising set of genetic programming techniques, including Multi Expression Programming (MEP) [9], Gene Expression Programming (GEP) [6], and Linear Genetic Programming (LGP) [5]. Although, some formulas programmed by MEP, GEP, and LGP have coefficients, such are all fixed constants [2]. Several researches employed GP derivatives for problems in construction industry. Baykasoglu et al. [3] employed GEP for concrete strength, cost and slump. Yeh and Lien [14] 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. Tsai [12,13] further proposed a weighted genetic programming (WGP) to introduce weight coefficients to tree connections and generates a fully weighted formula. This paper integrates GP and WGP as a genetic programming system (GPS) to investigate the feasibility for modeling deep beam strength.

The main purpose of this paper contributes on: 1) improvements and comparisons for GP and WGP, 2) accurate predictions and bonus formulas with the GPS, 3) modeling deep beam strength, and 4) attempts on polynomial-like formulas with GPS. The remaining sections of this paper include Section II: Proposed GPS and GA optimization; Section III: Modeling Deep beam strength with GPS; Section IV: Conclusions.

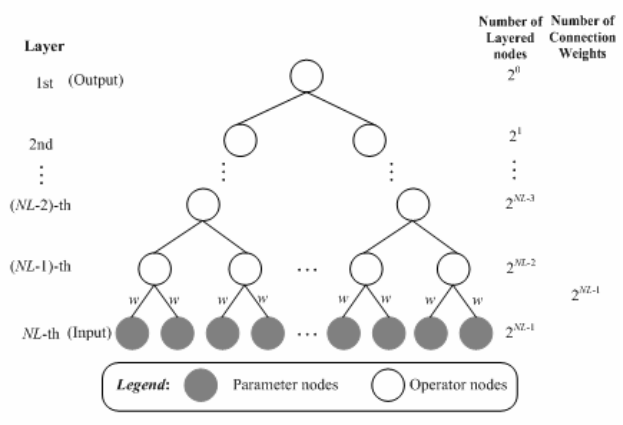

Fig. 1 Proposed Genetic programming structure.

2. Genetic Programming System

2.1 Genetic programming

This paper represents GP with a *NL*-layered tree structure as shown in Fig. 1. There are 2^{NL-1} parameter nodes in the eventual layer and each parameter node (x_i^{NL}) selects one of the inputs (including a unit parameter "1"). While a unit parameter is selected, the value of the parameter node adopts its attached weight (i.e. value at *w*) to create a coefficient. Therefore, final GP results are able to be equipped with optimized coefficients.

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

where x_i^{NL} represents nodes in the *NL*-th layer and *i* denotes a related node number; P_j is the *j*th input parameter; and *NI* is the number of inputs. Each x_i^{NL} node selects one attached P_j. Nodes in remained layers are operator nodes which use operators to calculate values of parent nodes in down-top order and are function of children nodes (see Fig. 2). Every operation node *y* is operated by a set of defined functions with two children nodal inputs of x_i and x_j.

$$y = F(x_i, x_j) = \begin{cases} f_0 = \quad \text{T} \\ f_1 = x_i + x_j \\ f_2 = x_i - x_j \\ f_3 = \quad x_i x_j \\ f_4 = \quad x_i / x \\ f_5 = \quad |x_i|^{x_j} \\ f_6 = \sin(x_i) \\ f_7 = \cos(x_i) \\ f_8 = \exp(x_i) \\ f_9 = \log|x_i| \end{cases} \tag{2}$$

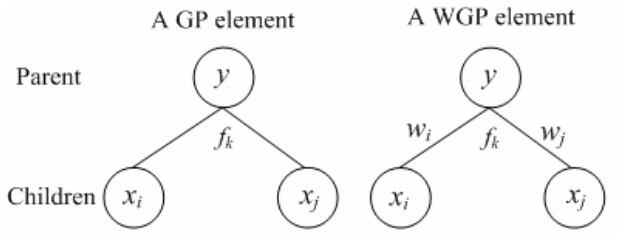

Fig. 2 Elements of GP and WGP.

This paper adopted f_0 and nine functions in eq. (2) for every operator selection *F*. The f_0 is a unique operator designed to a terminal of the branch (use "T" to represent). While "T" is selected for an operator node, it uses the value of the most left-hand side parameter node as its value to substitute for calculating its value from two children nodes. Therefore, some operator nodes do not appear in final GP results, while the "T" is used. Besides, the "T" is defined as a default function. f_1 is plus operator using "+" and f_2 is minus operator related to "-". f_3 and f_4 are multiply and divide operators related to "×" and "/" respectively. f_5 is a power("^") operator. The above five operator nodes are all binary nodes. f_6, f_7, f_8, and f_9 are all unary functional operators ("sin", "cos", "exp", and "log") and employ their left-hand children nodes.

2.2 weighted genetic programming

Although the aforementioned GP can produce

coefficients, such a coefficient costs (wastes) a branch of the GP structure (tree). Therefore, coefficients do not frequently happen in GP results. Tsai [12,13] introduced weighted balance for GP to create a WGP (see Fig. 3). Similarly, each parameter node x_i^{NL} selects one of the inputs (including a unit parameter "1") following eq. (3). While a unit parameter is selected in GP, its nodal value is a weight, but WGP uses one directly, owing to the weights are applied by operators already. Every operation node y is operated by a set of defined functions with two children nodal inputs of x_i and x_j involving weights of w_i and w_j.

$$y = F(w_i, w_j, x_i, x_j) = \begin{cases} f_0 = \mathrm{T} \\ f_1 = w_i x_i \\ f_2 = w_i x_i + w_j x_j \\ f_3 = (w_i x_i)(w_j x_j) \\ f_4 = (w_i x_i)/(w_j x_j) \\ f_5 = |w_i x_i|^{w_j x_j} \\ f_6 = \sin(w_i x_i) \\ f_7 = \cos(w_i x_i) \\ f_8 = \exp(w_i x_i) \\ f_9 = \log|w_i x_i| \end{cases} \quad (3)$$

The f_0 is the default "T". The f_1 is designed to inherit the left-hand side child nodes with w_i scaling and it is a unary operator (use "S" to represent). Such "S" is absent in GP function sets. f_2 is plus operator using "+". However "-" operator is absent in WGP, because of f_2 being able to cover such minus terms. f_3 to f_9 are "×", "/", "^", "sin" , "cos", "exp", and "log" with balanced weights.

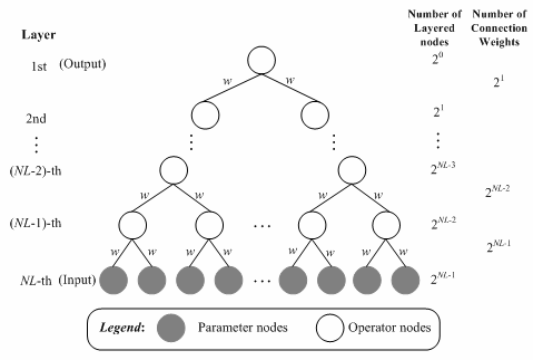

Fig. 3 Weighted Genetic Programming Structure.

All materials for GPS including both GP and WGP are considered ready with the exception of the operator selections, parameter selections, and weight optimizations.

2.3 Genetic Algorithm Optimization

This paper used the MATLAB *ga* function. The five basic steps to use GA are:

(1) Initialize Population - Initial individuals are randomly generated in types of "doubleVector" varied within 0~1, containing parameter selections (one x_i^{NL} node selects an attached P_j); operator selection (F) and weights (w). Values are linearly transferred to boundaries of x_i^{NL}, F, and w at integer 0~NI, integer 0~9, and float -10~10 respectively.

(2) Evaluate Individuals - The fitness function was directly set as inverse of the training root mean square error (RMSE). A larger fitness value indicates a healthier individual.

(3) Perform Crossover –This study performed a scattered (function of @corssoverscattered in MATLAB) crossover with a crossover rate of 0.8.

(4) Perform Mutation - The MATLAB function @mutationuniform with a mutation rate at 0.05 was used herein.

(5) Select Individuals - The @selectionstochinif was used herein to select parents of crossover and mutation children. In additional, two elitist individuals were guaranteed to survive to the next generation herein.

The population size chosen for this study was 200 and 5,000 iterations was adopted.

3. Modeling Deep Beam Strength

3.1 Data for deep beam Strength

Fig. 4 illustrates the problem of deep beam strength. Data used same as those in Tsai [11]. Ten input factors selected. Of 62 datasets, 52 were employed in training and 10 in testing (see Table 1). The best RMSE results of hybrid multilayer perceptrons are 8.3 and 8.2 kN in training and testing respectively [11]. This GPS study does not attempt to achieve results more accurate than those

above, but in modeling deep beam strength with formulas. Therefore, achieving a RMSE around 10 kN seems to be essential and good for GPS.

3.2 Modeling deep beam strength with GPS

This paper employed *NL* layers ranging from 2 to 6 for modeling deep beam strength. The selected model was judged by minimums of the summation of training and testing RMSE. The final GP^{NF}_{NL} formulas are listed as follows:

$$GP^{9}_{2} = P_3 P_7 \tag{6}$$

$$GP^{9}_{3} = \frac{P_6}{P_1}\sin(P_3) \tag{7}$$

$$GP^{9}_{4} = P_5^{P_1}\log|P_{10}| + P_3 P_7 + \frac{10}{P_4} \tag{8}$$

$$GP^{9}_{5} = (\frac{P_2}{P_8} + P_2 - P_{10})\cos(P_1 P_4) - \frac{2P_3 P_7}{P_5}(\log P_4 - \log P_{10}) \tag{9}$$

$$GP^{9}_{6} = \frac{P_2 + 8.15}{P_8 \exp(\frac{P_9}{P_5})} + \| P_9 - P_3 |^{\exp(P_3)}|^{\log(P_5 P_{10})} - P_5 + 2P_9 + P_7^{P_4} - P_2 + P_9^{P_{10}} - \frac{5.76}{P_4} - P_3 P_5 + \cos(P_5 P_8) \tag{10}$$

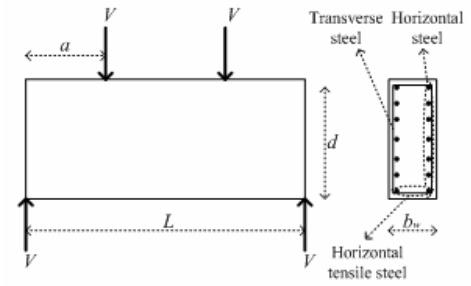

Fig. 4 Basic parameters and cross section for a deep beam.

No doubt, a higher *NL* provides a more complex formula and more accurate maybe. Basing on prediction accuracy and compact formulas, this paper suggested GP^{9}_{4} as a GP result for modeling deep beam strength. In GP^{9}_{4}, only P_1, P_3, P_4, P_5, P_7, and P_{10} participated in GP^{9}_{4}. This paper did not affirm that P_2, P_6, P_8, and P_9 have no impacts on deep beam strength. Under the batch of training data and accuracies at 12.8 / 11.2 kN, this paper selected GP^{9}_{4} for modeling deep beam strength with GP (also see Fig. 5). Different conditions may lead other scenarios. However, studying on all GP^{9}_{NL} results or finding more GP^{9}_{4} alternatives could conclude parameter impacts more detailed. Summarily, the GP in this paper has abilities for modeling problems with formulas against black box approaches; providing float coefficients against some GP derivatives; and being able to studying parameter impacts with programmed formulas.

Table 1 Lower and upper bounds for inputs and deep beam shear strength.

Factors	Lower Bound	Upper Bound
P_1: Effective span of beam (m), L	0.686	1.473
P_2: Breadth of beam (mm), b_w	76.2	101.6
P_3: Effective depth of beam (m), d	0.216	0.724
P_4: Shear span (m), a	0.254	0.635
P_5: Cylinder compressive strength of concrete (MPa), $f_c^{'}$	16.07	24.55
P_6: Yield strength of horizontal steel (MPa), f_{yh}	280	460
P_7: Yield strength of vertical steel (MPa), f_{yv}	280	460
P_8: Reinforcement ratio of horizontal tensile steel (%), ρ_h	0.52	1.94
P_9: Reinforcement ratio of total horizontal steel (%), ρ_{ht}	0.52	2.95
P_{10}: Reinforcement ratio of transverse steel (%), ρ_v	0.18	2.45
Ultimate shear strength (kN), V	87.40	249.1

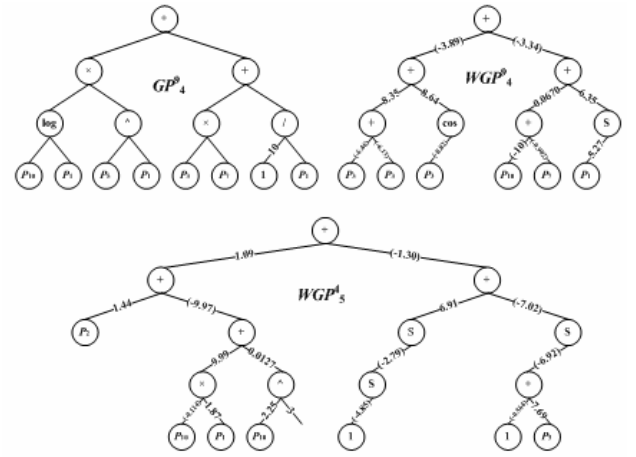

Fig. 5 Final GPS structures of the GP^{9}_{4}, WGP^{9}_{4}, and WGP^{4}_{5}.

Obviously, coefficients do not frequently happen in above GP results. WGP provides fully weighted formulas. Therefore, WGP will program formulas that totally different to those of GP. The final WGP^{NF}_{NL} formulas are listed as follows:

$$WGP^9{}_2 = 4.73P_2P_3 \quad (11)$$

$$WGP^9{}_3 = -0.902\frac{P_2}{P_1} + 8.87(3.81P_3)^{5.14P_8} \quad (12)$$

$$WGP^9{}_4 = 409P_3 - 33.6\cos(-8.82P_3) + 2.34P_{10} + 0.202P_7 - 112P_1 \quad (13)$$

$$WGP^9{}_5 = 38.5\log|0.107P_7 - 15.9| + 81.9\frac{P_4\cos(4.56P_1)}{P_1} + 9.94\log|8.58\exp(-8.83\cos(4.15P_3))| \quad (14)$$

$$WGP^9{}_6 = 24.5\log|-175P_1 - 175P_4 + 21.0P_5 + 139P_{10}| + \frac{100}{3.37^{2.2P_1}} + 5.37\frac{\exp(2.85\log|4.21P_3|)}{\log|7.94\sin(1.09P_3)|} - \frac{165P_2 + 242_{10}}{(6.39P_8)^{9.57P_3}} + 7.35\cos(9.40P_1) \quad (15)$$

In terms of compact formulas and good accuracies, $WGP^9{}_4$ was suggested as a WGP formula for modeling deep beam strength (also see Fig. 6). Within which, P_1, P_3, P_5, P_7, and P_{10} participated in $WGP^9{}_4$ providing accuracies at 12.7 / 12.4 kN under condition of the training data. Impact parameters were partially consistent with those in $GP^9{}_4$. Summarily, GP and WGP make GPS capable of accurate prediction, programming formulas, and parameter studies.

In GPS designs, functions for operator selection are determined by specific demands. Different settings for operators lead to various scenarios. This paper further designed a set of functions to provide polynomial formulas.

3.3 Modeling deep beam strength with WGP polynomials

Polynomials appear in a wide variety of areas of mathematics and science. Owing to GP is not capable of providing coefficients easily; constants and exponents for polynomials can not be produced frequently. This paper therefore used WGP to model deep beam strength in polynomials. WGP operators are redesigned with four functions as:

$$y = F(w_i, w_j, x_i, x_j) = \text{one}\begin{cases} f_0 = & \text{T} \\ f_1 = & w_i x_i \\ f_2 = & w_i x_i + w_j x_j \\ f_3 = & (w_i x_i)(w_j x_j) \\ f_4 = & (w_i x_i)^{w_j} \end{cases} \quad (16)$$

where f_1, f_2, f_3, and f_4 are "S", "+", "×", and "^", respectively; the exponent term w_j in f_4 uses integers at 1~10 to create positive integers as exponent terms for WGP polynomials. The final WGP polynomials are listed as follows:

$$WGP^4{}_2 = 4.74P_2P_3 \quad (17)$$

$$WGP^4{}_3 = 3.84P_2P_3 - 81.9P_1 + 98.7 \quad (18)$$

$$WGP^4{}_4 = 3.79P_2P_3 - 113P_4P_8 - 11.0P_5 - 398P_8 \quad (19)$$

$$WGP^4{}_5 = 1.56P_2 + 23.1P_1P_{10} - 1.58P_{10}{}^3 - 122P_1 + 35.6 - 486P_3 \quad (20)$$

$$WGP^4{}_6 = 366P_3 - 37P_3{}^2 - 85P_3{}^6 - 782P_2{}^2P_3 + 126P_3P_8 \quad (21)$$

Although, WGP polynomials were less accurate than those of $WGP^9{}_{NL}$, the programmed polynomials were more meaningful. Four observations should be addressed: 1) five-layered WGP polynomials seem enough for modeling deep beam strength in terms of accuracy; 2) $WGP^4{}_2$ is exactly same as $WGP^9{}_2$, a watershed appears while *NL* reaches three; 3) $WGP^4{}_4$ achieved a good accuracy at 13.7/10.5 kN; 4) results of $WGP^9{}_{NL}$ are varied in terms of parameter impacts; and 5) P_2 (breadth of beam) and P_3 (effective depth of beam) seems significant to WGP polynomials for programming deep beam strength especially in $WGP^4{}_2$, $WGP^4{}_3$, and $WGP^4{}_4$. It does make sense that deep beam strengths are impacted by a multiply of P_2 and P_3, i.e. the cross section area.

4. Conclusion

The proposed GPS integrates GP, WGP, and a WGP derivative (a WGP polynomial) for both predicting and programming deep beam strength. The GP is different with other GP approaches in providing constants by attaching weights for all parameter nodes. The WGP is continuous research by improving GP with fully weighted formulas. The WGP polynomial is designed for modeling problems in polynomial formulas. Significant findings of this paper include:

1. Improvements on GP emphasize on the use of attached weights for parameter nodes. Those on WGP improvements stick on employments of the "T" operator and the creation of WGP polynomials.
2. GPS approaches are capable of providing prediction results and programming results against common black-box soft computing approaches. GPS accuracies do not surrender to NN approaches while a big *NL* is used, but slightly deteriorate in concerns of compact formulas.
3. Studying GPS by layers brings accuracy improvements, accurate formulas, and impact parameters.
4. WGP polynomials complete a demonstration of function settings for operator selection and result in good polynomial formats. Other function settings for various problems may implement with concerns of target formula format or a lucky guess from good expertise.
5. Finally, GP^9_4, WGP^9_4, WGP^9_5, and WGP^9_4 are suggested for modeling deep beam strength. The former three achieve an optimal balance of prediction accuracy and compact formulas. The last WGP^9_4 reveals the evidence of modeling deep beam strength according to bases of beam cross section area.

References

[1] American Concrete Institute (ACI). *Building code requirements for reinforced concrete*. ACI 318-95, Detroit, 1995.

[2] Baykasoglu, A., Güllü, H., Çanakçi, H., Ozbakir, L., "Prediction of compressive and tensile strength of limestone via genetic programming", *Expert Systems with Applications*, Vol. 35(1-2), pp. 111-123, 2008.

[3] Baykasoglu, A., Oztas, A., Ozbay, E., "Prediction and multi-objective optimization of high-strength concrete parameters via soft computing approaches", *Expert Systems with Applications*, Vol. 36(3), pp. 6145-6155, 2009.

[4] Behzad, M., Asghari, K., Eazi, M., Palhang, M., "Generalization performance of support vector machines and neural networks in runoff modeling", *Expert Systems with Applications*, Vol. 36(4), pp. 7624-7629, 2009.

[5] Bhattacharya, M., Abraham, A., Nath, B., "A linear genetic programming approach for modeling electricity demand prediction in Victoria", *Proceedings of the hybrid information systems*, Adelaide, Australia, 379-393, 2001.

[6] Ferreira, C., "Gene Expression Programming: A New Adaptive Algorithm for Solving Problems", *Complex Systems*, Vol. 13(2), pp. 87-129, 2001.

[7] Koza, J.R., *Genetic programming: On the programming of computers by means of natural selection*, MIT Press, 1992.

[8] Mehrjoo, M., Khaji, N., Moharrami, H., Bahreininejad, A., "Damage detection of truss bridge joints using Artificial Neural Networks", *Expert Systems with Applications*, Vol. 35(3), pp. 1122-1131, 2008.

[9] Oltean, M., Dumitrescu, D., "Multi expression programming", UBB-01-2002, Babes-Bolyai University, Cluj-Napoca, Romania, 2002.

[10] Tsai, H.-C., "Hybrid High Order Neural Networks", *Applied Soft Computing*, Vol. 9(3), pp. 874-881, 2009.

[11] Tsai, H.-C., "Predicting Strengths of Concrete-type Specimens Using Hybrid Multilayer Perceptrons with Center-Unified Particle Swarm Optimization", *Expert Systems with Applications*, Vol. 37, pp. 1104-1112, 2010.

[12] Tsai, H.-C., "Using Weighted Genetic Programming to Program Squat Wall Strengths and Tune Associated Formulas", *Engineering Applications of Artificial intelligence*, Vol. 24, pp. 526-533, 2011.

[13] Tsai, H.-C., (2011), "Weighted Operation Structures to Program Strengths of Concrete-typed Specimens Using Genetic Algorithm", *Expert Systems with Applications*, 38, 161-168.

[14] Yeh, I.C., Lien, L.-C., "Knowledge discovery of concrete material using Genetic Operation Trees", *Expert Systems with Applications*, 36(3) 5807-5812, 2009.