Ocean Engineering 111 (2016) 8–12

Contents lists available at ScienceDirect

Ocean Engineering

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

Stability assessment of rubble-mound breakwaters using genetic programming

Mehmet Levent Koç [a,*], Can Elmar Balas [b,1], Dilek İmren Koç [c,2]

[a] Cumhuriyet University, Faculty of Engineering, Civil Engineering Department, 58140 Sivas, Turkey
[b] Gazi University, Faculty of Engineering, Civil Engineering Department, Celal Bayar Bulvarı, 06570, Ankara, Maltepe Turkey
[c] Cumhuriyet University, Faculty of Engineering, Chemical Engineering Department, 58140 Sivas, Turkey

ARTICLE INFO

Article history:
Received 4 May 2015
Received in revised form
4 September 2015
Accepted 30 October 2015
Available online 12 November 2015

Keywords:
Artificial intelligence
Rubble-mound breakwaters
Genetic programming
Stability assessment
Stability number

ABSTRACT

This study investigates the possibility of applying genetic programming to the stability assessment of rubble-mound breakwaters. The genetic programming algorithm is employed to develop various explicit models to predict the stability number of armor blocks. The genetic programming models are trained and tested using the experimental data of Van Der Meer (1988). The results show that the genetic programming based models have better predictive performances than Van der Meer's stability equations, and that genetic programming has good potential for solving complex problems in the field of coastal engineering.

1. Introduction

In the recent past, a number of studies have come up with predictive alternative methods in the field of stability analysis of rubble-mound breakwaters, such as artificial neural networks (ANNs) (Balas et al., 2010; Kim et al., 2008; Iglesias et al., 2008; Kim and Park, 2005; Mase et al., 1995), fuzzy neural networks (FNNs) (Koç and Balas, 2012) and model trees (MTs) (Shahidi and Bali, 2011). These studies generally show that (i) these artificial intelligence-based models are successful in the prediction of the stability numbers (ii) their forecasting ability is better than the traditional ones, e.g., Van der Meer's empirical models or stability equations (Van Der Meer, 1988) (iii) their application in the design of coastal structures requires continued research to explore the usefulness of alternative techniques, since they are relatively new compared to the conventional design practices. These are the reasons why the present work examines the applicability of another data-driven method, which is a genetic programming (GP), to assess the stability of rubble-mound breakwaters.

A GP (Koza, 1992) is a purely nonlinear modeling approach that can be described as an extension of well-known genetic algorithm (GA). The main difference between them is the representation of the solution. While a GA uses a string of numbers that represent the solution, GP solutions are computer programs. A GP creates computer programs to solve a problem using the principle of Darwinian natural selection. It mainly differs from other data driven models (e.g., ANNs, FNNs and MTs) in that it defines an explicit functional relationship between input and output variables by optimizing the model structure and its coefficients simultaneously. The applications of GP in coastal engineering are very few and include prediction of waves (Kambekar and Deo, 2012; Nitsure et al., 2012; Jain et al., 2011; Gaur and Deo, 2008) and sea water levels (Ghorbani et al., 2010).

The focus of this study is to develop explicit formulations based on the GP for accurately predicting the stability numbers of rubble-mound breakwaters. The GP models are trained and tested using the experimental data of Van der Meer (1988) and their predictive performances are compared with those of Van der Meer's stability equations. This study is the first to investigate the implementation of GP in this field.

2. Genetic programming

The process of GP starts with a random initial population of computer programs. An individual program present in the population refers to a parse tree (Fig. 1(a)), which is generated by the combination of its functions (nodes) and terminals (leaves) that are defined in a function set and terminal set, appropriate to the problem, respectively (Koza, 1992). A function set may consist of

* Corresponding author. Tel.: +90 346 2191010/1318; fax: +90 346 2191170.
E-mail addresses: mkoc@cumhuriyet.edu.tr (M.L. Koç), cbalas@gazi.edu.tr (C.E. Balas), dimren@cumhuriyet.edu.tr (D.İ. Koç).
[1] Tel.: +90 312 5833256; fax: +90 312 2308434.
[2] Tel.: +90 346 2191010/2242; fax: +90 346 2191170.

http://dx.doi.org/10.1016/j.oceaneng.2015.10.058
0029-8018/

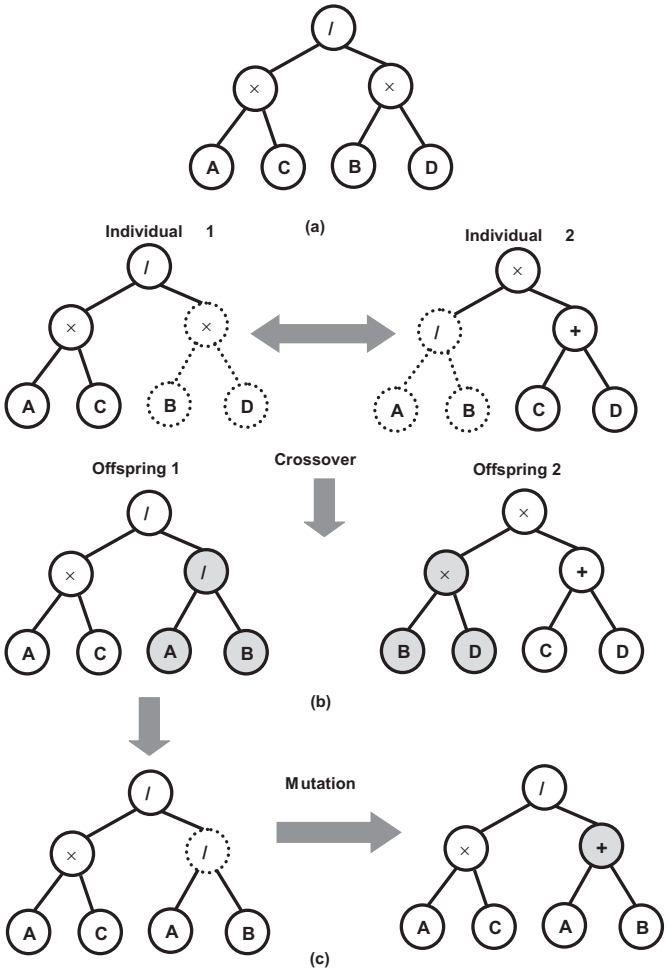

Fig. 1. An example of parse tree (a) generating $(A \times C)/(B \times D)$, crossover (b) and mutation (c).

basic arithmetic operators, mathematical functions, conditional operators, Boolean operators, iterative functions and any user-defined functions or operators, while a terminal set contains the arguments for the functions. Once the initial population has been created, the next step is repeatedly replacing the current population with a new population (or new generation) by means of applying genetic operators (reproduction, crossover and mutation) probabilistically until the best fitness of the population has reached the desired level, or the maximum number of generations has been reached. The genetic operators applied in a GP are the basic GA operators. The reproduction is the process of copying the selected individual program to the new population. The crossover operation creates a new offspring program for the new population by recombining randomly chosen parts from two selected programs (Fig. 1(b)). The mutation operator exchanges a randomly chosen part of one selected program in order to create one new offspring program for the new population (Fig. 1(c)). The best offspring program appearing in any generation, or the best-so-far solution, is able to solve the given problem in the best way (Koza et al., 2008).

3. Implementation of GP

The aim of this work was to develop GP models for exploring explicit relationships between the stability number of armor blocks and influencing variables. Van Der Meer (1988) showed that the formulation of the stability number (N_s) can be considered

Table 1
Range of variables in the training and testing subsets for Type 1 data set.

Design variables	Training data (207 data samples)	Testing data (372 data samples)
P	0.1000–0.6000	0.1000–0.6000
$\frac{S}{\sqrt{N}}$	0.0080–1.0318	0.0129–0.7930
ξ_m	0.6700–7.1500	0.7100–7.58000
$\cot\theta$	1.5000–6.0000	1.5000–6.0000
ξ_m^P	0.9607–2.8022	0.9663–2.8022
S	0.4400–32.6300	0.7100–32.9700
N	1000–3000	1000–3000
N_s	0.7900–4.3800	0.9500–4.1600

Table 2
Range of variables in the training and testing subsets for Type 2 data set.

Design variables	Training data (40 data samples)	Testing data (22 data samples)
P	0.5000–0.5000	0.5000–0.5000
$\frac{S}{\sqrt{N}}$	0.0136–0.8467	0.0186–0.3262
ξ_m	2.3400–4.9800	2.4300–4.9800
$\cot\theta$	2.0000–2.0000	2.0000–2.0000
ξ_m^P	1.5297–2.2315	1.5588–2.2315
S	0.7500–46.3800	0.6800–17.8700
N	1000–3000	1000–3000
N_s	1.4000–3.9700	1.4000–3.5500

to be as follows:

$$N_s = f\left(P^A, \left(S/\sqrt{N}\right)^B, \xi_m^C, \left(\xi_m^P\right)^D, (\cot\theta)^E\right) \quad (1)$$

where P is the permeability coefficient, S is the damage level, N is the number of waves, ξ_m is the surf similarity parameter, $\cot\theta$ is the slope of the breakwater, A, B, C, D and E are the constants to be determined that lie in the ranges between -1 and 1. Consequently, the experimental data of Van Der Meer (1988) were employed for generating various GP models relating to the N_s to the same input variables as Van der Meer's stability equations.

In the first stage, the experimental data set of Van Der Meer (1988) was divided into two main categories (Type 1 and Type 2 data sets): Type 1 data set included 579 data samples of small-scale tests on the static stability of rock slopes and large-scale tests on scale effects. Type 2 data set contained 62 data points of a low-crested structure. These labeled data sets were then used to train and test different GP models, as outlined below. The ranges of the design variables of randomly selected training and testing subsets for Type 1 and Type 2 data sets are given in Tables 1 and 2, respectively. The training sets were used in the training (or genetic evolution) phase, while the testing sets were used to measure the performance of the models obtained by the GP algorithm.

In the second stage, the various parameters involved in the GP algorithm were selected as follows: the size of the randomly generated initial population was set to 100. The function set was $F1 = \left\{\times, /, \exp, \log, \sqrt{}, \sqrt[3]{}\right\}$. The selection operator was implemented by using the tournament selection method (i.e., in which a specified group of programs are chosen at random from the population and the one with the better fitness is then selected) (Koza, 1992; Goldberg, 1989) to reproduce computational programs in proportional to the fitness values. The function to calculate the fitness (f) was expressed by the following equation:

$$f = \frac{100}{SSE} \quad (2)$$

where SSE is the sum of the squared errors for the training data set. The crossover and mutation operators were performed