Ocean Engineering 215 (2020) 107812

Contents lists available at ScienceDirect

Ocean Engineering

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

Genetic Programming for storm surge forecasting

Nguyen Thi Hien [a], Cao Truong Tran [a,*], Xuan Hoai Nguyen [b,1], Sooyoul Kim [c], Vu Dinh Phai [a], Nguyen Ba Thuy [d], Ngo Van Manh [d]

[a] Le Quy Don Technical University, 236 Hoang Quoc Viet St, Hanoi, Viet Nam
[b] Ho Chi Minh City University of Technology (HUTECH), Viet Nam
[c] Graduate School of Engineering, Tottori University, Japan
[d] Vietnam National Hydrometeorolocical Forecasting Center Hanoi, Viet Nam

ARTICLE INFO

Keywords:
Storm surge
Genetic programming
Typhoon
Surge deviation
White-box forecasting

ABSTRACT

Storm surge is a genuine common fiasco coming from the ocean. Therefore, an exact forecast of surges is a vital assignment to dodge property misfortunes and to decrease a chance caused by tropical storm surge. Genetic Programming (GP) is an evolution-based model learning technique that can simultaneously find the functional form and the numeric coefficients for the model. Therefore, GP has been widely applied to build models for predictive problems. However, GP has seldom been applied to the problem of storm surge forecasting. In this paper, we propose a new method to use GP for evolving models for storm surge forecasting. Experimental results on datasets collected from the Tottori coast of Japan show that GP can evolve accurate storm surge forecasting models. Moreover, GP can automatically select relevant features when evolving storm surge forecasting models, and the models evolved by GP are interpretable.

1. Introduction

Storm surge, a rise in sea level due to low atmospheric pressure and strong winds, is a severe natural disaster coming from the sea. Storm surges are especially harmful when they happen at a high tide, combining the impacts of the surge and the tide (Lee, 2008). With over 600 million individuals living in low-lying coastal zones, coastal surges have devastating societal impacts. For example, the most remarkable recorded storm surge in the United States was produced by Storm Katrina in 2005, which created a storm surge 9 m tall within the town of Cove St. Louis, Mississippi. The overall misfortune as a result of Katrina is evaluated to surpass $100 billion (Muis et al., 2016). Whereas high-impact occasions will, without a doubt, happen within the future, the advancement of the forecast of storm surge is able to enormously reduce the misfortune of lives and possibly lessen the sum of property harm (Kim et al., 2016, 2018; Thuy et al., 2016).

A conventional approach to storm surge prediction is to utilize process-based numerical forecasting models, but this approach is often computationally expensive. An alternative way is to use (data driven) machine learning algorithms such as artificial neural networks (ANNs) for predicting surge levels using the features such as sea surface levels, winds, sea level pressures, and tropical cyclone characteristics (Kim et al., 2016). For instance, Lee (2008) proposed a neural network combined with a consonant examination to forecast storm surges in Suao Harbor station, Taiwan. The experimental results showed that storm surge is effectively anticipated using neural systems. De Oliveira at al (De Oliveira et al., 2009). proposed a neural network model to predict ocean level varieties related to meteorological occasions. The results showed that the model is able to capture the impacts of the climatic and maritime intuitions. You at al (You and Seo, 2009). combined neural networks and clustering algorithms to build a storm surge prediction model. The observed results demonstrated that the model can be used for effectively forecasting territorial storm surges. Kim at al (Kim et al., 2016) demonstrated the impact of using different feature sets on an artificial neural network-based after-runner surge forecast model. The experimental results indicated that the combination of surge level, sea-level pressure, drop of sea-level pressure, longitude and latitude of typhoon, sea surface level, wind speed and wind direction comprises the optimal feature sets for predicting the surge level with the lead time of 24 h in the area of Sakai Minato on the Tottori coast, Japan.

Genetic programming (GP) is an evolutionary procedure to create solutions in the form of computer programs (Koza, 1992). The ability of

* Corresponding author.
E-mail address: truongct@lqdtu.edu.vn (C.T. Tran).
[1] Also at AI Academy Vietnam, 489 Hoang Quoc Viet St, Hanoi.

https://doi.org/10.1016/j.oceaneng.2020.107812
Received 24 February 2020; Received in revised form 16 July 2020; Accepted 17 July 2020
Available online 17 August 2020

GP to learn the definition of a function itself from sample information makes it a great choice for symbolic regression. Therefore, GP has been widely used to build regression models for many real applications. For instance, based on predictions of stock-prices using GP, a possibly profitable trading strategy was proposed in (Kaboudan, 2000). Gaur and Deo (2008) proposed the use of GP to build a model for real-time wave forecasting. The estimates created by GP shown that it can be respected as a promising tool for future applications to ocean forecasts. Azamathulla and Ghani (2010) utilized GP to anticipate river pipeline scour, and the execution of GP was found to be more viable when compared with the results of regression equations and artificial neural systems modeling in anticipating the scour depth around pipelines.

Recently, GP has also been applied to forecast storm surge. Sahoo and Bhaskaran (2019) proposed to use GP for predicting storm surge and inundation characteristics resulting from tropical cyclones. Experiments used datasets collected from the coast of Odisha adjoining the Bay of Bengal. Experimental results showed that both ANNs and GP perform very well as evidenced from their validation with actual data. However, GP has not been investigated to build models for storm surge forecasting with a lead time. Moreover, the ability of GP to automatically select features and build interpretable models for storm surge forecasting has not been studied. Therefore, this paper will investigate the ability of GP to build models for forecasting storm surge levels.

1.1. Goals

This paper aims to develop accurate and interpretable models for storm surge forecasting based on GP approach. The proposed method is compared with other common machine learning-based storm surge forecasting models to answer the following research questions:

1. Whether the GP-based storm surge forecasting models can be more accurate than machine learning-based storm surge forecasting models;
2. Whether GP can automatically select relevant features when building storm surge forecasting models.
3. Whether storm surge forecasting models evolved by GP are interpretable.

The rest of this paper is organized as follow. Section 2 gives a brief on GP and Machine Learning methods utilized in this paper. The proposed method and experiment design are shown in Sections 3 and 4. The results and analyses are displayed and examined in Section 5. Discussions are given in Section 6. Section 7 states conclusions and future work.

2. Backgrounds

2.1. Genetic Programming

Genetic Programming (GP) is one of the foremost well known evolutionary algorithm methods motivated by natural selection to evolve solutions, as computer programs, to problems (Koza, 1992, 1994). Within the 1990s, GP was primarily applied to mainly simple problems since it was rather computationally expensive. Nowadays, with the exponential development in CPU control and the changes of GP frameworks, it has broadly been utilized to solve various real-world problems. The applications of GP cover electronic design, diversion playing, quantum computing, and sorting (Koza, 1992, 1994).

Fig. 1 depicts the pseudo-code of GP. The following components are required for the application of GP (Poli et al., 2008).

2.1.1. Representation of candidate solutions

In spite of the fact that, for GP, there are a number of ways proposed to represent candidate solutions, the most popular one is tree-based representation. To produce a populace of individuals, firstly, a GP user selects a function set and a terminal set. The function set is regularly chosen from arithmetic operations (e.g +, -, *,/), mathematical functions (e.g., sin, cos, exp, log), Boolean operations (e.g., AND, OR, XOR, NOT) and conditionals (such as IF-THEN-ELSE). The terminal set frequently incorporates a number of features and a few constants. After that, a GP individual is built up recursively from the chosen functions and terminal sets (Koza, 1992; Poli et al., 2008).

2.1.2. Initializing a population

The reason for the initialization step is to initialize a population of seeding individuals that will be evolved in the later stages of evolution. In a tree-based GP framework, the *grow*, the *full*, and the *ramped half-and-half* strategies are the three primary methods for initializing the initial population.

To produce an individual in the *grow* initialization, firstly, a function within the function set is arbitrarily chosen and is considered as the root node of the tree. Expecting that the arity of the chosen function is *n*, at

Randomly make an starting populace of programs from the available primitives.

repeat

- Perform each program and evaluate its wellness.
- Select one or two program(s) from the population with a probability based on wellness to go to genetic operations.
- Apply genetic operations to make new individual program(s) with indicated probabilities.

until *halting condition is met (e.g., an satisfactory arrangement is found, or a most extreme number of generations is come to)*

return the best-so-far individual;

Fig. 1. Pseudo-code of genetic programming.

that point n nodes are randomly generated from the function and terminal sets as the children of the root node. In case a function is chosen, the recursive process is applied to that function. If a terminal is chosen, however, this branch of the tree is ended. The maximal depth of the tree is normally used to control the individual size. In *full* initialization, when building a tree, rather than choosing nodes from the function and terminal sets, only functions are chosen from the function set until it comes to the maximal depth where terminals are finally randomly selected from the terminal set.

For *ramped half-and-half* initialization, a half of the population is created by utilizing the *grow* strategy, while the other half is produced by the *full* strategy. This strategy is perhaps the most popular initialization procedure for GP (Poli et al., 2008).

2.1.3. Genetic operators

Crossover:. is the primary operator of GP. It produces new children that are created from parts embodied in each parent; hence, crossover makes variation within the population. For the execution of standard crossover, firstly, two parents are chosen according to a selection strategy, and after that one subtree is randomly chosen in each parent. If the two chosen subtrees are complied to the requirements (depth of the resultant children, syntactic closure property, etc.), the crossover operation is done by swapping them. At that point, the new offspring are included in the next generation (Koza, 1992).

Mutation:. is an asexual operator, i.e. it works with only one parent. To start, a mutation point is randomly chosen. After that the subtree rooted at the mutation point is expelled. A randomly created subtree is used to replace the outgoing one (Koza, 1992).

2.1.4. Fitness evaluation

Each individual within the population is assigned a numerical value called fitness measuring its capacity to solve the problem. Additionally, fitness must also be computational efficiency (Koza, 1994; Poli et al., 2008). Two frequently used fitness measures in GP are *raw* and *standardized* fitness. *Raw* fitness simply reflects the extent, to which an individual can solve the problem. *Standardized* fitness is calculated from the raw fitness so that the smaller fitness value, the better individual is in solving the problem (Koza, 1994; Poli et al., 2008).

2.1.5. Selection mechanism

Based on its fitness value, each individual in the population has a chance to be chosen for the breeding of the new population. The three most well known selection methods are *tournament*, *fitness proportionate* and *ranked* selection.

In *tournament* selection, a randomly chosen subset of individuals from the population is compared with each other in terms of fitness. At that point, the fittest is chosen to go to the mating pool. The flexibility of tournament selection lies in the choice for the size of the tournament subset. Normally, tournament selection does not require excessive fitness comparisons of all individuals. Subsequently, it may avoid expensive processing time and is easily subjected to parallelization (Koza, 1994; Poli et al., 2008).

In *fitness proportionate* selection, each individual is selected to the mating pool based on a likelihood that is proportionate to its fitness compared to the fitnesses of all individuals in the population. In spite of the fact that proportionate selection has been frequently utilized in GP and Genetic Algorithms, its behavior emphatically depends on the contrast between fitnesses of all individuals in the population (Koza, 1994; Poli et al., 2008).

In *ranking* selection, based on their fitness, all individuals are sorted. After that, based on the arranged order, each individual may be selected with a likelihood calculated from its rank (higher rank has higher chance to be chosen). Linear and exponential rankings are regularly utilized to index individuals. Although ranking selection strategy helps to reduce the shortcoming of proportionate selection, in a few cases, particularly within the exponential ranking, this strategy enlarges the contrasts between individuals with closed fitnesses; hence, the superior one might be chosen more frequently (Koza, 1994; Poli et al., 2008).

2.2. Machine learning methods

In this section, we will briefly portray four other machine learning methods tried out in our experiments. They are the Multi-Layer Perceptron (), k-Nearest Neighbors (k-NN), Decision Trees (DCT), and Support Vector Regression (SVR), which are ordinarily called data-driven models for the capacity to capture the mapping between input and output variables (forecast problems) without studying directly the natural rules that dominate the mechanism of storm surges. These models are completely based on information obtained from the collected data.

2.2.1. K-nearest neighbor

The k-NN is a non-parametric machine learning method that bases its forecast on the target yielded by the k-nearest neighbors of the given inquiry point (Song et al., 2017). In detail, given a data point, we determine the value of Euclidean measure between that point and all points within the training set. After that, we select its closest k nearest neighbors. Then we set the prediction value as the average of the target output values produced by these k nearest neighbors. In particular, the prediction x_t is defined as:

$$x_t^F = \frac{1}{k} \sum_{t \in S(X,m)} x_t \tag{1}$$

where $S(X, m)$ expresses the k-nearest neighbor index set to the $X(m)$ attribute vector. Intuitively speaking, the forecast x_t^F in Equation (1) is the sample average of the output surge level of the k-nearest neighbors to $X(m)$.

2.2.2. Support vector machines

Support vector machines can be utilized for both classification and regression problems. When applied to a regression problem it is called as support vector regression (SVR). In consideration of a linear model for illustration, the forecast is formulated by:

$$f(x) = w^T x + o, \tag{2}$$

where w, o and x denote the weight vector, the bias and the input vector, respectively. Let x_m be the m-th training input vector and y_m is the target output, $m = 1, \ldots, M$.

The following formula is used for the error function:

$$J = \frac{1}{2}||w||^2 + C \sum_{i=1}^{n} \left| y_m - f(x_m) \right|_\varepsilon \tag{3}$$

In this error function, the first term is used to penalizes the complexity of the model. The remaining term is called the ε-insensitive loss function, i.e. it does not penalize the errors below *epsilon*.

For the linear case, the learnt function could be obtained by the minimization of Eq. (4) as follows.

$$f(x) = \sum_{m=1}^{M} (\alpha_m^* - \alpha_m) x_m^T x + o \tag{4}$$

where α_m and α^*_m are Lagrange multipliers. The training vectors that provide nonzero Lagrange multipliers are referred to as support vectors, and that is a key concept in the theory of SVR. Non-support vectors indirectly contribute to the solution, and the number of support vectors is the degree of the complexity of the model (Balabin and Lomakina, 2011). This model can be extended to the non-linear case by kernelization with the introduction of the kernel function κ in a reproducing kernel Hilbert space (RKHS).

$$f(x) = \sum_{m=1}^{M} (\alpha_m^* - \alpha_m)\kappa(x_m^T x) + o \quad (5)$$

The widely used Gaussian kernel is employed in our experiments with SVR. Its width, δ_κ, is the standard deviation of the Gaussian function.

2.2.3. Multi-layer perceptron networks

The multi-layer perception network is by far the most common Artificial Neural Networks (ANN) model, which usually uses the error backward propagation technique to train the configuration of the network. The ANN architecture is made up of a number of hidden layers and a number of neurons in the input layer. In the output layer, either multiple or single neurons exist. ANNs with one hidden layer are usually utilized in hydrologic modeling (Dawson and Wilby, 2001; de Vos and Rientjes, 2005) on account that those networks are taken into consideration to provide sufficient complexity to accurately simulate the nonlinear properties of the hydrologic process. The ANN model for forecasting is defined as:

$$x_t^F = f(X_t, w, \theta, m, h) = \theta_0 + \sum_{j=1}^{h} w_j^{out} \varphi \left(\sum_{i=1}^{m} w_{ji} x_t + \theta_j \right), \quad (6)$$

where φ indicates the transfer function; w_{ji} is the weight of the link between the input layer's i-th node and the hidden layer's j-th node; θ_j is bias related to the j-th node of the hidden layer; w_j^{out} is the weight associated with the link between the j-th node of the hidden layer and the node of the output layer; and θ_0 is the bias at the output node. A suitable training algorithm is needed to find optimal w and θ in order to apply Equation (6) to surge level forecasting.

2.2.4. Decision Tree-REF tree

Reduces Error Pruning (REP) Tree Classifier is a decision-making tree learning algorithm based on the rule of splitting the training data using entropy based information gain and minimizing the error emerging from variance (Mohamed et al., 2012). REP Tree applies regression tree logic, and in altered iterations, it produces several trees. It chooses the best one of all spawned trees afterward. Then, it uses variance and knowledge gain to construct the regression/decision tree. The method also prunes the tree using a reduced-error pruning process that sorts numeric attribute values once at the start of the model preparation. As in the C4.5 algorithm (Quinlan, 2014), this algorithm also tackles the missing values by separating the related instances into parts.

3. Genetic Programming for storm surge forecast

3.1. Problem statement

In the current research, we used the remotely transmitted hourly recorded meteorological and hydrodynamic data from the observation stations on the Tottori coast of Japan, which are exactly the same Kim et al. (2016). There are three types of features used to predict storm surge level:

- meteorological parameters: wind speed (WS) (m/s), wind direction (WD) (degree), sea-level pressure (SLP) (hPa), and drop rate of sea-level pressure (DSLP) from average sea-level pressure at five sites. These parameters were collected from five local weather sites at Hamada, Matsue, Yonago, Ama, and Saigo operated by the Japan Meteorological Agency
- typhoon-featured parameters: longitude and latitude (LG and LT) (degree), central atmospheric pressure (CAP) (hPa), and highest wind speed (HWP) near the typhoon center (m/s). The typhoon-featured parameters are gathered from three typhoons of Maemi (2003), Songda 2004, and Megi 2004.
- hydrodynamic parameters: sea surface level (SSL) (m) (=the atmospheric tidal level + the surge level) and surge level (m) (=observed sea surface level - atmospheric tidal level) (SL)

The parameters mentioned above were taken as the input of the GP forecast model. The surge level is also used as the input parameter because the hourly surge level separated from a total sea surface level can be technically obtained from the stations and transferred to the present real-time operating system. In the output layer, the surge levels with the lead times of 5, 12 and 24 h are predicted.

In order to train and test the GP-based surge forecast model with the input and output parameter sets mentioned in the previous paragraph, a chain of surge level forecasting experiments were performed with the measurement collected during Typhoons Maemi 2003, Songda 2004, and Megi 2004 (Kim et al., 2016). Data sets were collected at the hourly interval for 168h for each event. The parameters obtained during Typhoon Maemi 2003 and Songda 2004, respectively, were used for training. The data length is appropriate to a relationship between input and output for the training phase. The parameters obtained during Megi 2004 were used for testing. Through the phases of training and testing, we developed the GP-based surge forecast models with 5, 12, and 24h lead times at Sakai Minato. It is noticed that all parameters were modified to have dimensional-less orders before training, resulting in falling within the range of [–1, 1] after rendering the parameters dimensional-less.

3.2. GP-based surge level forecasting

Here we outline a GP approach to predict the surge level. Several modifications have been made to the GP-based surge level forecasting, which is detailed in the following. All variables available within the data are included in the first set of elements in the terminal set. The second element is an ephemeral random constant (ERC) that will select a random number that is uniformly distributed. In [–1, 1], we allow our ERC to select a random number. The set of functions includes: Add (ADD), Subtract (SUB), Multiply (MUL), Divide (DIV), square root (SQRT), natural logarithm (base e) (LOG) and trigonometric functions (sin, cos). LOG, SQRT, and DIV functions are protected since zeroes and negative numbers are included in the results. If the input is zero or negative, zero will be returned by SQRT and LOG. If zero (denominator) is the second argument passed to DIV, zero is returned as well. Protecting these values will prevent the generation of NaN's (not a number) and Inf (infinity). We make sure that the first child is either a function or a variable when initializing the population using the ramped-and-half, whereas the second child can either be a variable, an ERC or another function. This will prevent random numbers from dominating trees. Table 1 summarizes all the functions and terminals provided in this section.

The fitness used for assessment will be based on the root mean squared error and the coefficient of correlation (CC) given by:

Table 1
GP parameter settings.

Parameter	Value
Function set	+, -, x,/(protected division), sin, cos, SQRT, LOG
Variable terminals	all features
Constant terminals	Random float values
Population size	1024
Initialization	Ramped half-and-half
Generations	50
Crossover probability	60%
Mutation probability	30%
Reproduction rate	10%
Selection type	Tournament (size = 7)

$$fitness = NRMSE\left(1 + \frac{1}{CC}\right) \tag{7}$$

where *NRMSE* and *CC* are calculated by:

$$NRMSE = \frac{\sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(y_{obs,i} - y_{pre,i}\right)}}{\left(y_{obs,max} - y_{obs,min}\right)} \tag{8}$$

$$CC = \frac{\sum_{i=1}^{n}\left(y_{obs,i} - \bar{y}_{obs}\right)\left(y_{pre,i} - \bar{y}_{pre}\right)}{\sqrt{\sum_{i=1}^{n}\left(y_{obs,i} - \bar{y}_{obs}\right)^2 \sum_{i=1}^{n}\left(y_{pre,i} - \bar{y}_{pre}\right)^2}} \tag{9}$$

where n is the size of the training set, $y_{pre,i}$ represents the predicted surge level and $y_{obs,i}$ represents the actual surge level for the ith data point (time index).

Every search process starts with a random set of trees being generated. Every tree's objective function, shown in Equation (7), is then determined. Trees with better target values are picked using methods such as tournament selection. Two genetic operators are planning the chosen trees for the next iteration: crossover and mutation. The new trees that were produced are the input for the next iteration. This process continues to the maximum number of iterations or error satisfaction (e.g. equal zero).

4. Parameter settings

In all experiments, we used the implementation of GP in the ECJ library (A Java-based Evolutionary Computation Research System) (Luke et al., 2006). Table 1 shows the parameters of our GP systems. Thirty independent runs of GP were performed for each experiment, yielding diverse solutions at each run. These independent runs have different seeds generated by the computer pseudo-random number generator. We also utilized elitism in our evolutionary process, which copied the best individual of the current population to the next generation without changes.

In machine learning, regression learning algorithms are frequently categorized into decision trees such as C4.5 (Quinlan, 2014), instance-based classifiers such as k nearest neighbor (kNN) (Weinberger et al., 2006), and function-based classifiers such as an artificial neural network (ANN) and SVM (Han et al., 2011). Hence, four regression algorithms (C4.5 kNN, ANN, and SVM) were utilized in our experiments for the sake of comparison. We used the implementations of these regression learning methods in the WEKA library (Hall et al., 2009). The parameter settings of these algorithms are given in Table 2.

5. Results and analyses

5.1. Empirical evaluation of the proposed method

In this experiment of storm surge forecast, it was found that the difference in performance between GP with the number of generations was significant, as seen in Fig. 2(a) and (b), and 2(c). We choose values of the number of generations in [50, 500].

First, the correlation coefficient of surge forecast with 5h-lead time is examined. In Fig. 2(a), GP achieves the best performance with the maximum number of generations of 200. It seems that, in this case, the more number of generation number leads to the decrease of the correlation coefficient value. Next, in Fig. 2(b), with 12h-lead time, the correlation coefficient value increases rapidly to 200 then decreases gradually. In the last case with 24h-lead time, in Fig. 2(c), after reaching the peak (at $200 - th$ generation), the difference between the maximum value and the rear values is relatively small. Experimenting with different maximum number of generations (50, 60, to 500), we noticed that patterns of results for GP in general show the best correlation coefficient value in the case of 200 generations. For this reason, we set this parameter (maximum number of generations) to 200 in all subsequent experiments.

Table 2
Optimal parameters using weka for the four models: SVM, k-NN, ANN and C4.5 for surge level forecasting.

SVM		k-NN	
SVM Type	epsilon-SVR	k	7
Cost	9.6974	distance Function	Euclidean
Gamma	6.8399		
Kennel type	RBF		
Esilon	0.001		
ANN		C4.5	
hiddenLayers	5	minimum number of instances	2
learningRate	0.3	unpruned	no
momentum	0.2	number of folds	3
epochs	500		

5.2. The comparison between the proposed method and the machine learning methods

In this section, we look at the predictive error and coefficient correlation value between algorithms. Four aforementioned machine learning models are chosen to compete with GP. In addition, we also compare GPs to the best ANN forecast models in (Kim et al., 2016) on each data set. We compare the results of each algorithm on 3 data samples. GP and ANN is in the form of a stochastic algorithm, so we test GP for 30 times before taking the median result from the set of the best individuals on the training set from each run. SVM, k-NN, and DCT are deterministic, and only one run was required.

5.2.1. Accuracy of surge forecasts with the lead time of 5h

For 5h-lead time, the results in Kim et al. (2016) indicated that the ANN-based surge model (ANN-B3), which were trained with a combination of storm surge, sea-level pressure, drop rate of sea-level pressure, longitude, latitude, sea surface level, and wind speed, is the best performing model. Firstly, the accuracy of surge estimate with the 5h-lead time is assessed by comparing time series of the surge level and perceptions amid Tropical storm Megi as appeared in Fig. 3.

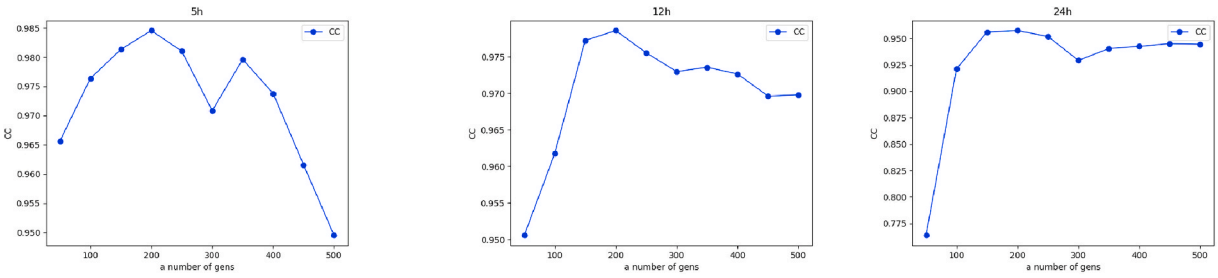
Taking a look at the testing results of the models in Fig. 3, the surge level outputs by ANN-B3, GP, and ANN are fairly good in comparison with the observations. However, we can find among these models, GP and k-NN achieved the best performance around the peak.

5.2.2. Accuracy of surge forecasts with the lead time of 12h

In this subsection, all forecast models for the lead time of 12h are assessed. In Kim et al. (2016), they found that the ANN-based surge model (ANN-B1) is the best model trained with features of storm surge, sea-level pressure, drop rate of sea-level pressure, longitude, and latitude. As appeared in Fig. 4, the 12h lead time forecasts of SVM (SVR) and ANN-B1 generally overestimated the surge levels, whereas GP and ANN predicted the surge levels with thin deviations. Among them, kNN and DCT appear jutted surge levels close to the greatest surge level.

5.2.3. Accuracy of surge forecasts with the lead time of 24h

Lastly, we assess the accuracy of the estimate models with the lead time of 24h. Time series of the surge figures and perceptions for Typhoon Megi are shown in Fig. 5. The ANN-based surge model (ANN-B4) is the best ANN model in Kim et al. (2016), which was trained with storm surge, sea-level pressure, drop rate of sea-level pressure, longitude, latitude, wind speed, wind direction. Overall, the forecasts of surge levels are rather inaccurate for all models. However, GP and ANN produced better forecasting values and relatively captured the trend of the data. By contrast, the models built by DCT and kNN are more fluctuated. For SVM and ANN-B4, it can be seen from Fig. 5 that they produced bad forecasts and failed to capture the data trend.



(a) 5h-lead time (b) 12h-lead time 2 (c) 24h-lead time 2

Fig. 2. The differentiation of correlation coefficient values with the number of generations on 3 data sets.

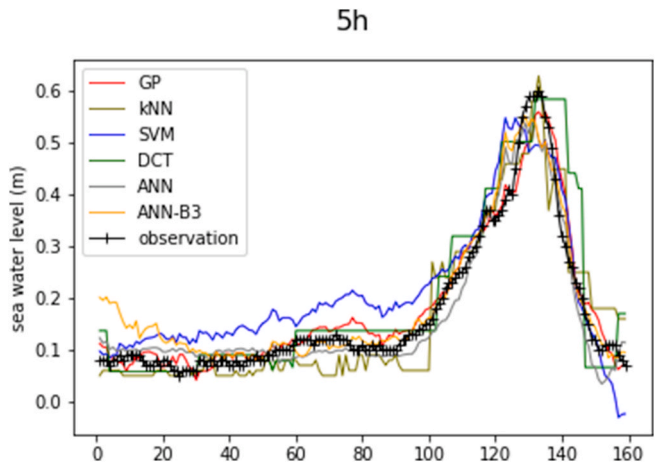


Fig. 3. Time series of observed and forecasted surge levels for Typhoon Megi with the lead time of 5h at Sakai Minato by 6 models.

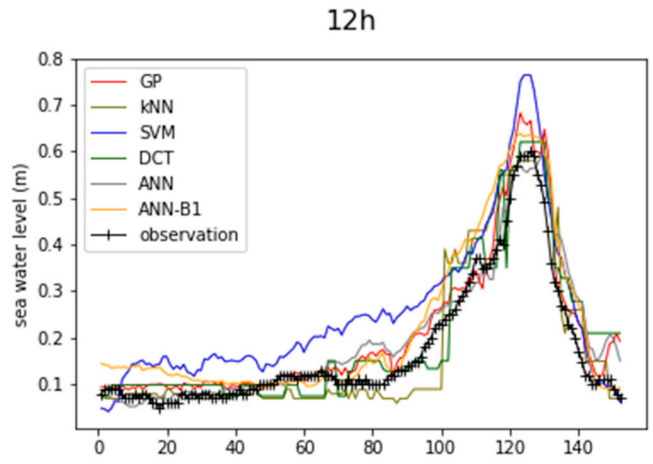


Fig. 4. Time series of observed and forecasted surge levels for Typhoon Megi with the lead time of 12 h at Sakai Minato by 6 models.

6. Discussions

6.1. Discussion on accuracy

To evaluate our evolved GP models in detail on the 5h-lead time forecasts compared to the other five learning methods, we calculated two records of the correlation coefficient (CC) and the normalized root mean square error (NRMSE) in rate as appeared in Fig. 6(a). It is observed that the value of CC in all of the models ranges from 0.93 to 0.99, while that of NRMSE is from 5% to 12%. Among these models, the one evolved with GP has smallest error and the largest correlation coefficient.

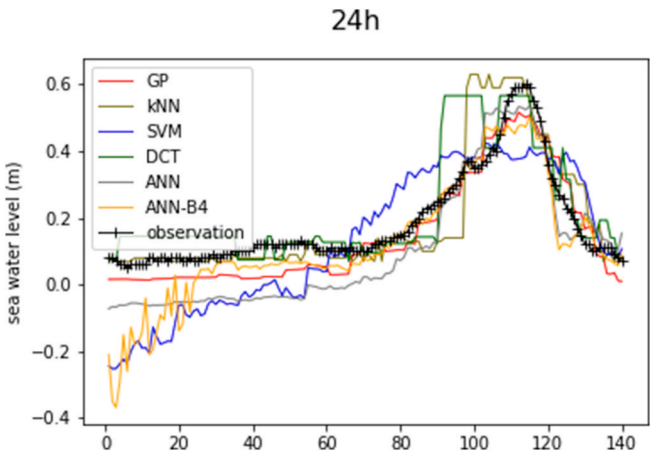
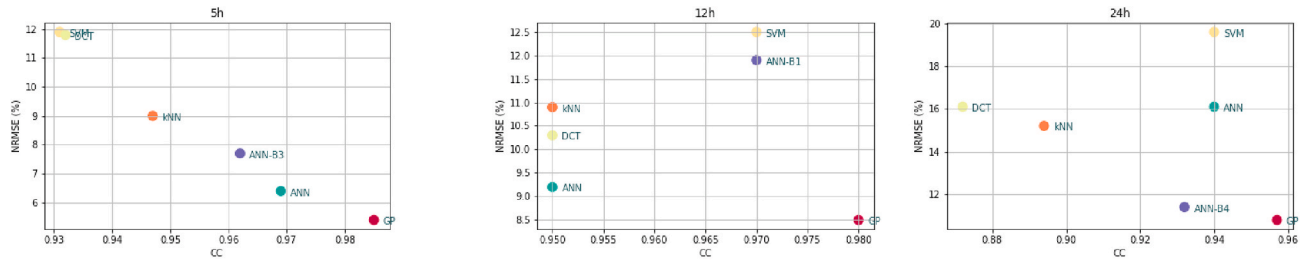


Fig. 5. Time series of observed and forecasted surge levels for Typhoon Megi with the lead time of 24 h at Sakai Minato by 6 models.

Similarly, in Fig. 6(b), the correlation coefficient (CC) and the normalized root mean square error (NRMSE) of all models for 12h-lead time surge level forecast are depicted. The CC values of the models mostly concentrated in the range of 0.94–0.98. These models also have the NRMSE values from 8% to 13%. Among them, GP shows the best performance for 12h-lead time forecasts.

Finally, two records of CC and NRMSE are again calculated for all models in 24h-lead time forecasts. They are shown in Fig. 6(c). The CC values of most of the models are scattered from 0.86 to 0.96 and the NRMSE values are from 10% to 20%. Even though, the performance of GP based model deteriorates compared to the 6h- and 12h-lead time cases, it is still the best forecasting model among all with the values of CC and NRMSE as 0.96 and 10.8%, respectively. Overall, these experiments and comparisons show that GP is the most effective method in building surge level forecasting model.

To further confirm the superiority of GP evolved models, we performed multiple statistical tests by using Friendman's test (Demšar, 2006) on NRMSE. Table 3 shows the rank of the algorithms by using Friedman's test (smaller means better). It is obvious from Table 3 that GP is the best algorithm, followed by ANN, and others. The Friedman's test shows that there exits significant differences between the methods. Therefore, we carry out the Holm test (Demšar, 2006) to perform pair tests between the two methods. With 5h-lead time, the Holm test shows that GP is significantly better than other methods except ANN. With 12h-lead time, GP is not significantly better than kNN, ANN_B1 and SVM. With 24h-lead time, GP is not significantly better than DCT, ANN and SVM.



(a) 5h-lead time (b) 12h-lead time (c) 24h-lead time

Fig. 6. Diagram of the correlation coefficient (CC) and the normalized root mean square error (%) for the surge level forecast made in 3 interval times in advance by 6 models.

Table 3
Ranking the algorithms by Friedman's test (smaller means better).

5h-lead time		12h-lead time		24h-lead time	
Algorithm	Ranking	Algorithm	Ranking	Algorithm	Ranking
GP	2.742	GP	2.815	GP	2.567
ANN	2.830	ANN	2.838	ANN_B4	2.875
ANN_B3	3.279	DCT	3.102	kNN	3.050
kNN	3.484	kNN	3.384	DCT	3.300
DCT	3.764	ANN_B1	3.875	ANN	4.453
SVM	4.899	SVM	4.983	SVM	4.753

In summary, storm surge forecasting models evolved by GP are generally more accurate than models built by the machine learning algorithms.

6.2. Discussion on the evolved models

The best models evolved by GP are shown in Table 4, where variables $x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9$, and x_{10} stand for the attributes (features): SSL numeric, SLP numeric, DSLP numeric, WS numeric, WD numeric, LT numeric, LG numeric, CAP numeric, HWS numeric, SS numeric, respectively.

It is clear from Table 4 that GP can automatically select features to evolve the best models. The model for the 5h-lead time forecasting uses four features: sea surface level, sea-level pressure, drop rate of sea-level pressure, and storm surge. The model for the 12h-lead time forecasting uses five features: sea-level pressure, drop rate of sea-level pressure, wind speed, longitude, and latitude. The model for the 24h-lead time forecasting only uses three features: sea-level pressure, longitude and latitude, and central atmospheric pressure.

It can be seen from the best model for the 5h-lead time forecasting that the incorporation of the storm-feature parameter like the typhoon position is able to improve the accuracy of the forecasting model. Moreover, training the model with the local parameters can forecast the storm surge with the 5h lead time, and adding the storm-feature parameter can improve the accuracy of the model.

With the best model for the 12h-lead time forecasting, the local parameters with the typhoon position are not enough to describe the storm surge with the 12 h lead time. Furthermore, adding the typhoon intensity, which is explained by the central atmospheric pressure, to the local parameters with the typhoon position, the storm surge with the 12h lead time can be accurately predicted.

It also can be seen from Table 4 that the best model for the 24h-lead time forecasting combines the meteorological and typhoon-feature parameters. In the 24h forecast models, the hydrodynamic parameters are excluded for the 24h forecasting in contrast to the 5h and 12h forecast models.

In summary, GP can automatically select features when evolving models, and the models evolved by GP is explainable.

Table 4
The simplified equations derived by GP for the surge level forecast.

Dataset	Equation
5 h-lead time	$sin\sqrt{sin(sin(sin(sinx_{10}) \times x_2)) \times \sqrt{x_{10} * x_3}} + x_1$
12 h-lead time	$x_4 + \sqrt{\frac{x_3 \times log\sqrt{logx_6}}{log(cos(x_7 + \sqrt{x_3} \times x_3^2) + x_3) \times x_2^2}}$
24 h-lead time	$x_8 \times log\sqrt{log\left(cos\frac{x_2}{x_6} \times x_6\right)}$

7. Conclusions and future work

This paper has proposed a method to use Genetic Programming (GP) for building storm surge forecasting models. Meteorological, hydrodynamic and typhoon-featured parameters were taken into Genetic Programming to build models forecasting storm surge levels with the lead times of 5h, 12h and 24h. The proposed method was evaluated on the datasets collected from the observation stations on the Tottori coast. The experiments compared Genetic Programming with other common machine learning methods on the normal root mean squared error and the coefficient of correlation. The experimental results showed that not only Genetic Programming can achieve than smaller error, but also achieve higher coefficient of correlation than other machine learning methods. Statistical testing further confirmed that storm surge forecasting models evolved by GP are significantly better than support vector machines, k nearest neighbor and decision tree. Moreover, Genetic Programming can automatically select relevant features when evolving storm surge forecasting models, and storm surge forecasting models evolved by Genetic Programming are interpretable (in closed-form equations).

Although this paper used Genetic Programming to build and test models on datasets from Tottori coast, the proposed method is independent with these datasets from Tottori coast. Therefore, the proposed method will be applicable to build storm surge forecasting models for other coasts.

In order to evolve more accurate models, the future work could investigate advanced Genetic Programming such as semantic Genetic Programming (Vanneschi et al., 2014) to evolve storm surge forecasting models.

CRediT authorship contribution statement

Nguyen Thi Hien: Methodology, Writing - original draft, Visualization. **Cao Truong Tran:** Methodology, Writing - original draft, Writing - review & editing, Formal analysis. **Xuan Hoai Nguyen:** Supervision, Validation, Writing - review & editing. **Sooyoul Kim:** Supervision, Validation, Writing - review & editing. **Vu Dinh Phai:** Data curation. **Nguyen Ba Thuy:** Resources, Funding acquisition. **Ngo Van**

Manh: Resources, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

The research presented in this paper was funded by National Science and Technology Program to respond to climate change, manage natural resources and the environment in the period 2016–2020 in the project titled "Applications of Artificial Intelligence for Forecasting Hydro-Meteorological Anomalies in Vietnam under the Context of Climate Change", Grant Number: BĐKH.34/16-20.

References

Azamathulla, H.M., Ghani, A.A., 2010. Genetic programming to predict river pipeline scour. J. Pipeline Syst. Eng. Pract. 1 (3), 127–132.
Balabin, R.M., Lomakina, E.I., 2011. Support vector machine regression (svr/ls-svm)–an alternative to neural networks (ann) for analytical chemistry? comparison of nonlinear methods on near infrared (nir) spectroscopy data. Analyst 136 (8), 1703–1712.
Dawson, C.W., Wilby, R.L., 2001. Hydrological modelling using artificial neural networks. Prog. Phys. Geogr.: Earth Environ. 25 (1), 80–108.
De Oliveira, M.M., Ebecken, N.F.F., De Oliveira, J.L.F., de Azevedo Santos, I., 2009. Neural network model to predict a storm surge. J. Appl. Meteorol. Climatol. 48 (1), 143–155.
de Vos, N., Rientjes, T., 2005. Constraints of artificial neural networks for rainfall-runoff modelling: trade-offs in hydrological state representation and model evaluation. Hydrol. Earth Syst. Sci. 9 (1–2), 111–126.
Demšar, J., 2006. Statistical comparisons of classifiers over multiple data sets. J. Mach. Learn. Res. 1–30.
Gaur, S., Deo, M., 2008. Real-time wave forecasting using genetic programming. Ocean. Eng. 35 (11–12), 1166–1172.
Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H., 2009. The weka data mining software: an update. ACM SIGKDD Explorations Newslett. 11, 10–18.
Han, J., Pei, J., Kamber, M., 2011. Data Mining: Concepts and Techniques. Elsevier.
Kaboudan, M.A., 2000. Genetic programming prediction of stock prices. Comput. Econ. 16 (3), 207–236.
Kim, S., Matsumi, Y., Pan, S., Mase, H., 2016. A real-time forecast model using artificial neural network for after-runner storm surges on the tottori coast, Japan. Ocean. Eng. 122, 44–53.
Kim, S.-W., Lee, A., Mun, J., 2018. A surrogate modeling for storm surge prediction using an artificial neural network. J. Coast Res. 85 (sp1), 866–870.
Koza, J.R., 1992. Genetic Programming: on the Programming of Computers by Means of Natural Selection, vol. 1. MIT press.
Koza, J.R., 1994. Genetic programming as a means for programming computers by natural selection. Stat. Comput. 4 (2), 87–112.
Lee, T.-L., 2008. Prediction of storm surge and surge deviation using a neural network. J. Coast Res. 24 (sp3), 76–82.
Luke, S., Panait, L., Balan, G., Paus, S., Skolicki, Z., Bassett, J., Hubley, R., Chircop, A., 2006. Ecj: a java-based evolutionary computation research system. Downloadable versions and documentation can be found at the following url. http://cs.gmu.edu/eclab/projects/ecj.
Mohamed, W.N.H.W., Salleh, M.N.M., Omar, A.H., 2012. A comparative study of reduced error pruning method in decision tree algorithms. In: 2012 IEEE International Conference on Control System, Computing and Engineering. IEEE, pp. 392–397.
Muis, S., Verlaan, M., Winsemius, H.C., Aerts, J.C., Ward, P.J., 2016. A global reanalysis of storm surges and extreme sea levels. Nat. Commun. 7, 11969.
Poli, R., Langdon, W.B., McPhee, N.F., Koza, J.R., 2008. A Field Guide to Genetic Programming. Lulu. com.
Quinlan, J.R., 2014. C4. 5: Programs for Machine Learning. Elsevier.
Sahoo, B., Bhaskaran, P.K., 2019. Prediction of storm surge and inundation using climatological datasets for the indian coast using soft computing techniques. Soft Comput. 23, 12363–12383.
Song, Y., Liang, J., Lu, J., Zhao, X., 2017. An efficient instance selection algorithm for k nearest neighbor regression. Neurocomputing 251, 26–34.
Thuy, N.B., Kim, S., Chien, D.D., Dang, V.H., Cuong, H.D., Wettre, C., Hole, L.R., 2016. Assessment of storm surge along the coast of central vietnam. J. Coast Res. 33 (3), 518–530.
Vanneschi, L., Silva, S., Castelli, M., Manzoni, L., 2014. Geometric semantic genetic programming for real life applications. In: Genetic Programming Theory and Practice Xi. Springer, pp. 191–209.
Weinberger, K.Q., Blitzer, J., Saul, L.K., 2006. Distance metric learning for large margin nearest neighbor classification. In: Advances in Neural Information Processing Systems, pp. 1473–1480.
You, S.H., Seo, J.-W., 2009. Storm surge prediction using an artificial neural network model and cluster analysis. Nat. Hazards 51 (1), 97–114.