Genetic Improvement of Shoreline Evolution Forecasting Models

Mahmoud Al Najar*
CNRS/CNES/ISAE-SUPAERO
Toulouse, France
mahmoud.al.najar@legos.obs-mip.fr

Rafael Almar
The French Research Institute for Development (IRD)
Toulouse, France
rafael.almar@ird.fr

Erwin W. J. Bergsma
Earth Observation Lab, The French Space Agency (CNES)
Toulouse, France
erwin.bergsma@cnes.fr

Jean-Marc Delvit
Earth Observation Lab, The French Space Agency (CNES)
Toulouse, France
jean-marc.delvit@cnes.fr

Dennis G. Wilson
ISAE-SUPAERO, University of Toulouse
Toulouse, France
dennis.wilson@isae-supaero.fr

ABSTRACT

Coastal development and climate change are changing the geography of our coasts, while more and more people are moving towards the coasts. Recent advances in artificial intelligence allow for automatic analysis of observational data. Cartesian Genetic Programming (CGP) is a form of Genetic Programming (GP) that has been successfully used in a large variety of tasks including data-driven symbolic regression. We formulate the problem of shoreline evolution forecasting as a Genetic Improvement (GI) problem using CGP to encode and improve upon ShoreFor, an equilibrium shoreline prediction model, to study the effectiveness of CGP in GI in forecasting tasks. This work presents an empirical study of the sensitivity of CGP to a number of evolutionary configurations and constraints and compares the performances of the evolved models to the base ShoreFor model.

CCS CONCEPTS

• **Applied computing** → *Environmental sciences*; • **Computing methodologies** → *Representation of mathematical functions*; **Genetic programming**; **Genetic algorithms**.

KEYWORDS

symbolic regression, cartesian genetic programming, forecasting, shoreline evolution

ACM Reference Format:
Mahmoud Al Najar, Rafael Almar, Erwin W. J. Bergsma, Jean-Marc Delvit, and Dennis G. Wilson. 2022. Genetic Improvement of Shoreline Evolution Forecasting Models. In *Genetic and Evolutionary Computation Conference Companion (GECCO '22 Companion), July 9–13, 2022, Boston, MA, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3520304.3534041

*Corresponding author

GECCO '22 Companion, July 9–13, 2022, Boston, MA, USA

ACM ISBN 978-1-4503-9268-6/22/07…$15.00
https://doi.org/10.1145/3520304.3534041

1 INTRODUCTION

Coasts around the globe are continuously facing natural and anthropogenic pressures. Our knowledge and understanding of the evolution of the coastal zone over time is crucial for a large variety of applications including coastal risk monitoring and management. Shoreline evolution forecasting is an important element in coastal studies that aims to better understand and predict the occurrence and intensity of erosive and accretive forces. Recently, large efforts have been made to understand and predict shoreline evolution due to the rising social, economic and natural pressures such as climate change [3, 13, 14, 16]. Shoreline change occurs at varying time scales resulting from different natural processes, ranging from small oscillations resulting from individual waves to decadal trends as a response to varying wave climates. At seasonal to interannual scales, cross-shore sediment transport is considered the main driver of shoreline change, while alongshore processes are more relevant at longer timescales [10, 20, 21].

Three main types of methods have been proposed and discussed in the literature on the topic of forecasting shoreline change [13]. Process-based models are based on including as many physical simulations as possible which contribute to the process of shoreline change. These simulated processes are usually coupled through mass and momentum conservation laws. Hybrid models are mixed approaches to modelling shoreline change incorporating general physical principles, such as the principle of shoreline equilibrium [27], and are calibrated using data-driven approaches (e.g. least-squares-fit). Finally, data-driven techniques range from simple regression methods to modern deep learning models, which have demonstrated impressive capabilities over a wide variety of applications in many domains including physics [1] and coastal science [6, 8].

This work makes use of Recurrent Cartesian Genetic Programming (RCGP) [24] as a method for Genetic Improvement (GI). A forecasting model of shoreline change, ShoreFor [4, 20], is implemented as a CGP individual and evolved using a standard Genetic Algorithm (GA). The current work presents an ablation and sensitivity study of RCGP for GI in the context of time series analysis according to a number of evolutionary configurations and constraints. The layout of this article is as follows. We first describe the different methods used in this work in Section 2, including the shoreline equilibrium model ShoreFor which is used to initialize our

CGP population, the dataset and the task to be performed, Cartesian Genetic Programming and the methodology we follow in order to transform a system of equations into a CGP individual. Then, in Section 3, we present the different experiments that were conducted and we analyze their results. We present in Section 4 an example evolved graph and compare it to the base CGP-ShoreFor graph. In Section 5, we discuss the implications of the results and the possible future directions. Finally, we summarize our contributions and conclusions in Section 6.

2 METHODS

2.1 ShoreFor

ShoreFor [4, 20] is a shoreline change forecasting model that is built upon the principle of shoreline equilibrium [27], where shorelines continuously evolve towards a time-varying equilibrium condition. ShoreFor can be formulated according to Equation 1, where dx/dt is the rate of shoreline change, F is the magnitude of wave forcing, c and b are model free parameters that are optimized using a least-squares-fit minimizing the root-mean-squared-error (RMSE) of the model.

$$\frac{dx}{dt} = c(F^+ + rF^-) + b \tag{1}$$

The wave forcing term F (Equation 2) is expressed in terms of the wave energy flux P and the normalized disequilibrium term $\Delta\Omega/\sigma_{\Delta\Omega}$.

$$F = P^{0.5}\frac{\Delta\Omega}{\sigma_{\Delta\Omega}} \tag{2}$$

ShoreFor defines the beach equilibrium state (Ω_{eq}, Equation 3) as a weighted average of antecedent dimensionless fall velocities where ϕ is a model-free parameter which controls the number of days in the series used to estimate the current equilibrium state.

$$\Omega_{eq} = \frac{\sum_{i=1}^{2\phi} \Omega_i 10^{-i/\phi}}{\sum_{i=1}^{2\phi} 10^{-i/\phi}} \tag{3}$$

Ω, calculated according to Equation 4, represents the rate of sedimentation and is a function of the sediment grain settling velocity w, the breaking wave height $H_{s,b}$ and wave period T_p.

$$\Omega = \frac{H_{s,b}}{wT_p} \tag{4}$$

Disequilibrium, $\Delta\Omega = \Omega_{eq} - \Omega$, is used to partition forcing F into accretion and erosion (F^+, F^-) according to the sign of $\Delta\Omega$. The erosion ratio r (Equation 5) is defined as a ratio between the detrended accretive and erosive wave forcing. It is calculated over the full calibration set and treated as a constant to balance the accretion and erosion terms within the ShoreFor model.

$$r = \left| \frac{\sum_{i=0}^{N} \langle F_i^+ \rangle}{\sum_{i=0}^{N} \langle F_i^- \rangle} \right| \tag{5}$$

ShoreFor has been used in multiple shoreline studies and a number of extensions have been proposed to improve its performance by accounting for shoreline change over different time-scales [18] as well as alongshore sediment transport processes [22, 23]. We therefore make use of the ShoreFor model as a base for our experiments on the use of CGP for shoreline forecasting in a GI setting, with the eventual goal of extending our base CGP-ShoreFor implementation to account for these additional processes.

2.2 Dataset and Task

The dataset used in this work was obtained from the Shoreshop shoreline modelling competition [13]. The site studied is Tairua Beach located on the eastern coast of New Zealand. The dataset used in this work includes 15 years of hourly (Δt = 3 hours) video-derived shoreline positions and wave forcing time series including the wave height H_s and period T_p. We reduce the dataset to a daily scale (Δt = 24 hours) using a simple daily average of shoreline position and wave forcing. The dataset is split into 11 year and 4 year subsets, used to evaluate fitness during evolution and to test the evolved models. We refer to the original work in [13] for a more comprehensive description of the dataset and the physical characteristics of the study site.

In this work, we make use of a modified version of the Mielke skill test [5] in order to evaluate the fitness of the CGP individuals during evolution. Correlation-based metrics, in contrast with error-based metrics, are generally considered more informative in shoreline studies as they evaluate the trends captured by the model rather than the errors in model predictions. The Mielke skill test (Equation 6) is an extension of the Pearson correlation coefficient which measures the linear dependence between two datasets. It also accounts for any bias observed in the data to downgrade the similarity score and is therefore considered a more accurate estimate of model performance.

$$\lambda = 1 - \frac{N^{-1}\sum_{i=1}^{N}(o_i - m_i)^2}{\sigma_o^2 + \sigma_m^2 + (\hat{o} - \hat{m})^2} \tag{6}$$

We use this score to compare the output of a CGP individual to the ground truth data of shoreline position over the 11 year training period.

2.3 Cartesian Genetic Programming

Cartesian Genetic Programming (CGP) [11] is a form of genetic programming that encodes programs as directed acyclic graphs. A CGP graph is composed of three main components, the input nodes, output nodes and computation nodes. To represent a program as a genome, each node in the graph can be associated with three integers corresponding to the function of the node and its two inputs. More recent applications of CGP make use of a single vector of nodes to represent a complete program (1 row and N columns) [12]. A CGP genome is of fixed size, however the phenotype of a CGP graph can be of variable size as some nodes in the graph can be independent or disconnected from the output computation chain of nodes. This allows for flexibility in the number of nodes an individual can use, where nodes that do not contribute to the output are simply ignored during individual evaluation. CGP has been successfully applied to a large number of problems including digital circuit design [2], image processing [9, 19], computer vision-based applications [26], among others [12]. Here, we use a recurrent version of CGP (RCGP) [24] which has shown promising performance over sequential datasets compared to CGP [25]. RCGP can be considered as a superset of traditional CGP that allows for the

creation of cyclic connections in the computation graph according to a recurrent connection rate parameter.

2.4 ShoreFor Graph Implementation

In order to encode the ShoreFor system of equations as a single CGP genome, we manually decompose each of its equations into a dictionary representation, where each node has a unique key and three fields indicating the node's elementary function and both of its inputs. In addition to two additional lists indicating the unique keys of the inputs and outputs. For example, in the generalized version of ShoreFor [20], the weight vector in Equation 3 is computed such that the weighting factor decreases to 10%, 1%, and 0.1% at ϕ, 2ϕ, and 3ϕ days prior to present day ($W = 10^{-i/\phi}$). In order to encode the computation of this weight vector, two inputs and five different functions are required. The inputs include the Ω time series of the current timestep (Ω, where $length(\Omega) = 2\phi$) and the calibrated ϕ constant. Given these inputs, we decompose the calculation of W as follows. First, a vector of length 2ϕ with values ranging from 1 to 2ϕ is computed using the *irange* function. This vector is then simply flipped using the *reverse*, to represent the number of days back in history each point in the time series represents. At this point, the vector of i in $10^{-i/\phi}$ is computed. Then, this vector is negated using the *negate* function and divided by the input constant *phi* using *div*, resulting in a vector of values representing $-i/\phi$. This vector is passed to the *tpow* function ($tpow(x) = 10^x$), obtaining $W = 10^{-i/\phi}$. Table 1 summarizes the different categories of functions used in this work to create a pool of 50 candidate functions representing CGP's function set during evolution. The final set of inputs used in our CGP-ShoreFor implementation is as follows:

(1) Ω_t
(2) P_t
(3) ϕ
(4) 0.5
(5) 0.5
(6) Δt
(7) t_i
(8) $coeff_1$
(9) $coeff_2$
(10) r

Inputs (4) and (5) are currently included as constants in the inputs. This design choice was made with the eventual goal of evolving them and the ShoreFor calibrated parameters (inputs 3, 8 and 9). Input (4) is used in Equation 2, whereas inputs (5, 6, 7, 8, 9) are used at the end of the genome to compute the integral of Equation 1 such that the output of the CGP individual corresponds to the value of X (the cross-shore shoreline location).

Generally speaking, implementations of CGP require that all input and computed variables are bound to a range of -1 to 1 in order to prevent various computational issues such as the existance of NaN's or infinities in the computational graph. However, this requirement is difficult to achieve in the case of GI of a physical system of equations due to the lack of true maxima for each input and the use of unbounded functions in the original model. Therefore, we instead choose to handle out-of-bounds computation by penalizing all such individuals by assigning them a fitness value of negative infinity, essentially discarding them from future generations.

After encoding the ShoreFor system of equations as a single CGP genome, the ShoreFor individual can be represented as a graph structure as shown in Figure 1.

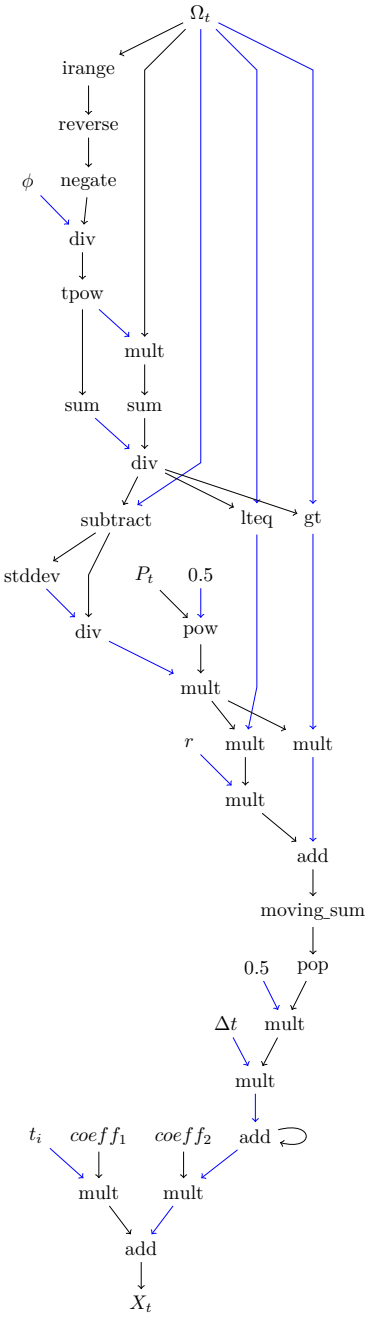


Figure 1: ShoreFor encoded as a CGP graph.

Currently, the implementation of ShoreFor as a CGP individual includes a modification to the original model formulation. The original ShoreFor model makes use of the full calibration time series in order to calibrate its different parameters (r, ϕ, c, b). The first modification to the ShoreFor model was to move from a full regression-based model that makes use of the full dataset, to an

Table 1: A description of the different types of functions included in the function set used for this work.

Type	*Description*
List processing	list manipulation functions (e.g. tail, pop, reverse)
Scalar	Scalar operations (e.g. add, div, mult)
Vector	Vector operations (e.g. sum, mean, max)

instantaneous version of ShoreFor (CGP-ShoreFor) that operates on 2ϕ days of inputs only. This simplification requires that the parameters (r, ϕ, c, b) are included as inputs to the CGP-ShoreFor model.

Moreover, our current implementation assumes that the equations to calculate the breaking wave power P and the dimensionless fall velocity Ω are physical fact and are therefore not included in the evolvable CGP-ShoreFor implementation, but rather passed as pre-calculated time series.

3 EXPERIMENTS AND RESULTS

This section describes four different experiments that were conducted and presents their results. For each experiment, the results are analyzed according to the modified Mielke skill score as a measure for fitness as presented in Section 2.2. Table 2 summarizes the default configuration parameters used in the ablation study. The active graphs of the individuals in the initial population correspond to the CGP-ShoreFor model, while the non-active nodes of each individual are initialized randomly. During evolution, a constrained version of the Goldman mutation [7] (constraints described in Section 3.2) is used to mutate the input, output or function genomes of an individual's computational node using a node mutation rate of 0.1.

3.1 Evaluation series length

The first experiment in this study is focused on minimizing the computation time required to evaluate the different CGP-ShoreFor individuals during evolution. As CGP-ShoreFor individuals are processed at each time step in the evaluation period, the evaluation period length has a direct correlation with the total evaluation time of an individual. To this end, we test minimizing the duration of the evaluation period. Table 3 and Figure 2 present the results obtained from this experiment.

Table 2: Default configuration

Parameter	*Value*
Evaluation series length	11 years
Number of columns	300
Constraints	all
Population size	30
Mutation rate	0.1
Output mutation rate	0.3
Recurrent connection rate	0.1

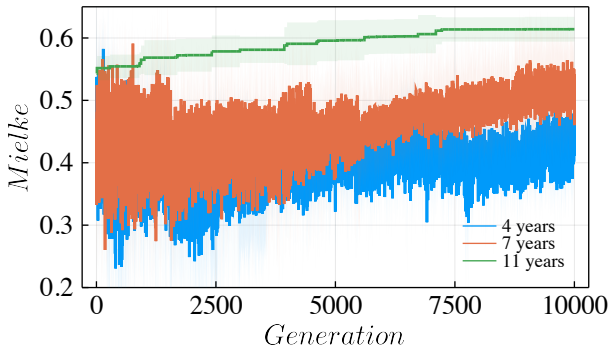


Figure 2: Comparison of the fitness evolution when using evaluation periods with varying sizes.

In both cases of 4 years and 7 years, the use of a smaller evaluation period led to a large reduction in computation time. However, the model performance varied greatly in between different parts of the time series when evaluated on a smaller time series, leading to a large level of noise in the performance signal during evolution. Interestingly, although noisy, the models evaluated on 7 years seem to be converging towards a better performing area of the search space which could lead to more global solutions. Based on the results presented in Table 3 and Figure 2, the individuals in the following experiments were evaluated on 11 years of data during evolution in order to have a more stable estimation of model performance.

3.2 Constraints

In this experiment, two types of constraints were implemented and tested to compare their effect on the convergence of the model population during evolution. These constraints can be split into two levels: evaluation constraints and mutation constraints. Evaluation constraints in this context correspond to constraints based on the statistical characteristics of the predicted time series compared to the original time series. Mutation-level constraints are those enforced during the mutation step of evolution, before evaluation. The constraints tested in this work build upon previous work that aims to reduce the computational overhead by skipping needless evaluations [7]. Table 4 first summarizes the different constraints that were added to the evolutionary method. Then, Table 5 presents a statistical overview of the performances of each constraint regime.

Table 3: Comparison of the average fitness and the standard deviation in fitness of the five top performing individuals from each set of experiments.

	fitness
4 years	0.46 ± 0.06
7 years	0.51 ± 0.06
11 years	0.61 ± 0.06

Figure 3 then showcases the effects of the different types of constraints on the method's speed of convergence.

Table 4 summarizes the different constraints tested. Two free parameters (α_1 and α_2) are introduced and used to implement these constraints. α_1 is used to evaluate the standard deviation of the model-produced shoreline evolution predictions and was set to 10% of the target standard deviation. If the constraint is not met, the model would get a negative infinity as its fitness value, essentially discarding it from following generations.

Additionally, three mutation-level constraints were tested. The first is concerned with the length of the active graph of the evolved individual, where any mutation resulting with an individual whose active graph has a length of less than α_2 is considered invalid. In our case, α_2 was set to approximately half the size of the ShoreFor graph used to initialize our individuals at the beginning of each run ($\alpha_2 = 20$). Second, we explicitly enforce that a mutation-produced model produces unequal outputs when applied to two random but different inputs. Finally, any mutated individual with a direct input-output connection as part of its active graph is considered invalid and a new mutation is created.

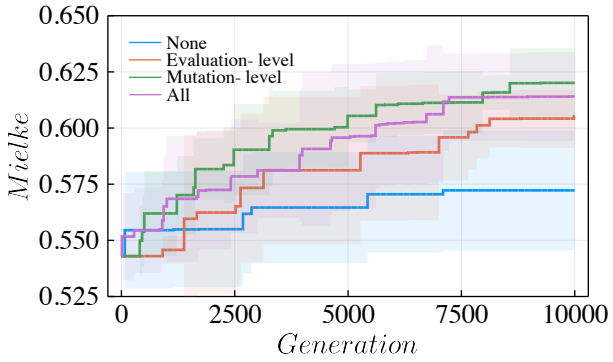


Figure 3: Comparison of model convergence according to different constraint regimes

Table 4: The different constraint regimes tested.

Level	*Constraint*
Evaluation	$std(\hat{y}) > \alpha_1$
Mutation	model length $> \alpha_2$
Mutation	$model(rand_1)\ != model(rand_2)$
Mutation	no input-output connections

Table 5: Comparison of the constraint regime's impact on final model performance.

	fitness
None	0.57 ± 0.03
Evaluation	0.61 ± 0.01
Mutation	0.62 ± 0.02
All	0.61 ± 0.02

Table 5 and Figure 3 showcase the impact of using different levels of constraints on the performance of CGP. The benefit of using constraints is shown numerically in Table 5, where the constrained runs all achieved greater than 0.6 performance on the Mielke skill test on average, while the non-constrained models achieve $\simeq 0.57$ only. Moreover, Figure 3 demonstrates the effect of contraints on the convergence of the method when the contraints are enabled during mutation and/or evaluation, where constrained models are able to achieve higher fitness scores earlier in evolution, in addition to producing more skilled models at the end of evolution.

3.3 Graph size

This subsection explores the impact of varying the size of the complete initial graph on CGP's ability to produce better-performing models. Table 6 first presents these results statistically using the top performing individuals from 5 separate runs. Figure 4 then showcases these differences over consecutive generations.

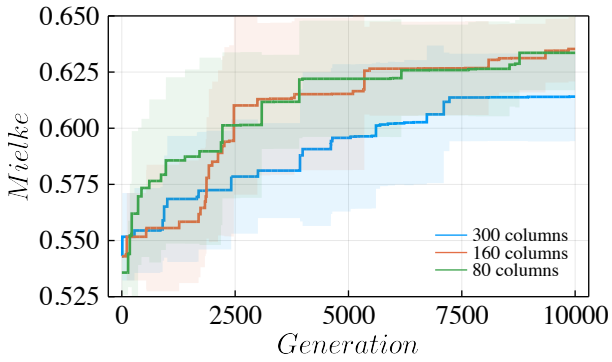


Figure 4: Model performance comparison using different CGP graph sizes.

The results shown in Table 6 and Figure 4 show that the use of a smaller graph size aids evolution in finding better-performing genomes. An important point to mention here is the expected complexity of the function to be evolved; even though smaller models seem to reduce the number of generations required for CGP to improve the initial model due to the reduced search space, using a small graph reduces the number of possible function representations that CGP can produce. In the context of this work, where CGP is used to essentially evolve a system of equations that can model shoreline change, without prior knowledge of the complexity of the physical system to be modelled, we argue that the use of larger model could be more beneficial.

Table 6: Final fitness comparison according to different CGP graph sizes.

	fitness
300	0.61 ± 0.02
160	0.64 ± 0.01
80	0.63 ± 0.02

3.4 Graph Initialization

In this section, we compare the ability of CGP to evolve a shoreline forecasting model starting from random genomes against the use of ShoreFor as a starting point for evolution. These results are presented in Table 7 and Figure 5

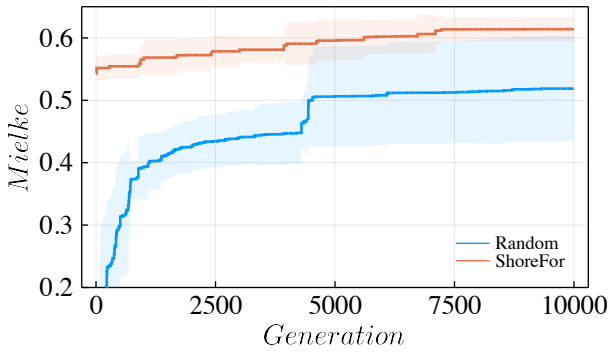


Figure 5: Comparing the convergence and performance of CGP-ShoreFor and pure CGP.

Table 7 compares the fitnesses of the evolved models when the problem is formulated as a pure CGP problem (random initialization) or as a GI problem using ShoreFor to initialize the population to be evolved. As shown in Table 7 and Figure 5, both methods achieve good performance in a relatively small number of generations (10,000), demonstrating the effectiveness of CGP in time series forecasting tasks. On average, the CGP-ShoreFor initialized models achieved significantly higher skill scores compared to the pure-CGP models which varied greatly depending on their initialization over the 5 different runs.

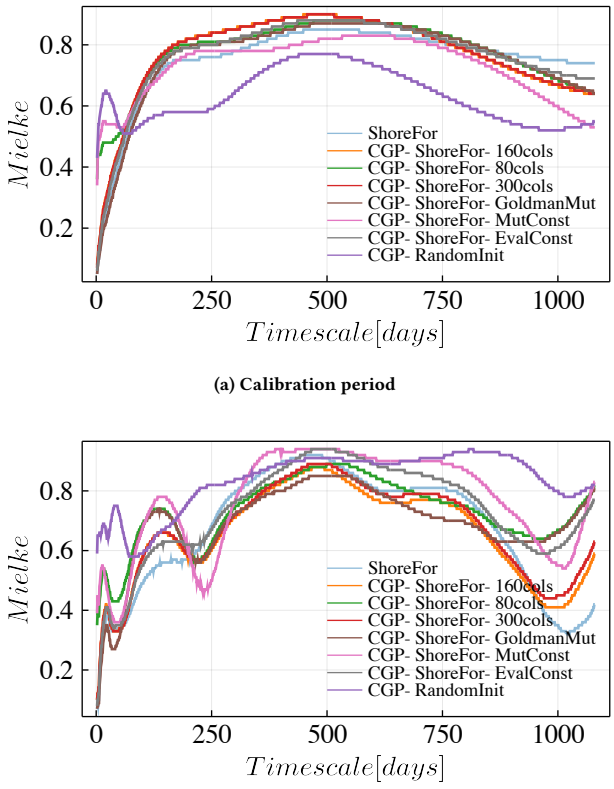
3.5 Performance Comparison

Figure 6 presents the Mielke skill scores of the top performing individuals from each experiment when evaluated over different timescales by using a pass-band filter to isolate trends of different temporal scales. This shows that the majority of the evolved versions are able to improve on the original ShoreFor model over seasonal to interannual time scales.

A comparison of the different CGP-ShoreFor variants shown in Figure 6 suggests that they achieve similar performance over most time scales. Interestingly, the pure-CGP model (the model with random initialization) appears to achieve the best performance over very short time scales during the calibration period (11 years) and better performance over multiple time scales during the forecast period (4 years). On the other hand, most of the CGP-ShoreFor

Table 7: Comparison of the final fitness performances of CGP-ShoreFor and pure CGP.

	fitness
Random	0.49 ± 0.09
ShoreFor	0.61 ± 0.02



(a) Calibration period

(b) Forecast period

Figure 6: Model performance comparison at different time scales over the calibration (a) and forecast (b) periods.

models improve on ShoreFor during the calibration period and are competitive with ShoreFor over different time scales during the forecast period, despite the fact that no additional calibration was performed on the input parameters described in Section 2.4. Figure 6b demonstrates the ability of the evolved models to achieve higher skill compared to the base model over previously-unseen inputs, suggesting that the evolved equations are applicable to different periods and are not a result of further fitting to a specific dataset. We note that the individuals used in this comparison correspond to the best performing individuals among 5 different runs from each experiment; we refer to Section 3.4 for a more comprehensive comparison between the randomly initialized models and the models initialized as CGP-ShoreFor.

4 EVOLVED SHOREFOR GRAPHS

In this section, we present one CGP-ShoreFor model resulting from evolution and we decode part of the evolved graph to better understand the modifications produced by evolution. Additional example evolved graphs are presented in Appendix A (Supplementary Material).

Figure 7 presents an example graph from an evolved individual. In the figure, we highlight inside green circles two differences

between this evolved individual and the original ShoreFor individual's graphs (Figures 7 and 1). The first difference concerns the computation of the weights in the calculation of Ω_{eq}. As described in Section 2.1, the original ShoreFor formulation calculates Ω_{eq} as a weighted mean of the historical dimensionless fall velocities within a specified period of time (Equation 3). As part of the Ω_{eq} calculation, the weights are computed as $W = 10^{-i/\phi}$, corresponding to a decaying weight coefficient. This computation takes place at the beginning of the ShoreFor graph (Figure 1). In the evolved graph, as highlighted in the top green circle in Figure 7, the weights at a specific point in time are computed as $W = 10^{-2\phi/i}$.

The second difference concerns the calculation of the forcing term F (Equation 2). Originally, the F term is calculated as a function of the breaking wave energy flux $P^{0.5}$ and the instantaneous normalized disequilibrium term $\Delta\Omega/\sigma_{\Delta\Omega}$. In the evolved version, F can be decoded as: $F_i = P_i^{0.5} * (\Delta\Omega_i + \Delta\Omega_{i-1})$, as a function of the breaking wave energy flux $P^{0.5}$ and the two last disequilibrium ($\Delta\Omega$) terms, suggesting that the model is relying heavily on very recent wave activity to produce its estimations.

Although the graphs such as 7 may be difficult to interpret at first, careful analysis enables us to understand slight modifications made to the ShoreFor model through Genetic Improvement. On the other hand, the resulting model can in some cases be significantly different from the original model (example graphs in supplementary materials, Appendix A), making the task of decoding the evolved models non-trivial. This is a strong motivation for further development of the algorithm, for example by incorporating multiple criteria in the fitness calculation to account for the modularity and/or complexity of the evolved graphs. The results presented in this section demonstrate the advantage of data-driven GI, as the resulting models are directly comparable to the source model but are more performant on the objective dataset.

5 DISCUSSION & FUTURE DIRECTIONS

This work made use of the CGP algorithm in a Genetic Improvement formulation of the problem of forecasting shoreline change using an established model to shoreline forecasting named ShoreFor. Multiple experiments were conducted to study the sensitivity of CGP to the different evolutionary configurations, with the aim of maximizing its ability to improve on its ShoreFor initialization.

We have demonstrated the effects of using different time series lengths during the evaluation step of evolution. The results shown in Figure 2 demonstrate that the noise in the evaluation measure increases as the evaluation duration is increased. This result is a strong motivation for further study in different noisy fitness estimation schemes [15, 17] due to the great potential reduction of computational resources required for the application of CGP in similar problems.

Furthermore, we have studied the impacts of integrating different constraints at the evaluation and mutation stages of evolution. A future direction of this work is to split the mutation constraints to study their impacts independently from one another in order to better understand their respective effects on evolution. We also aim to integrate different physical laws as evaluation-level constraints in order to enforce that all evolved individuals are physically-sound solutions.

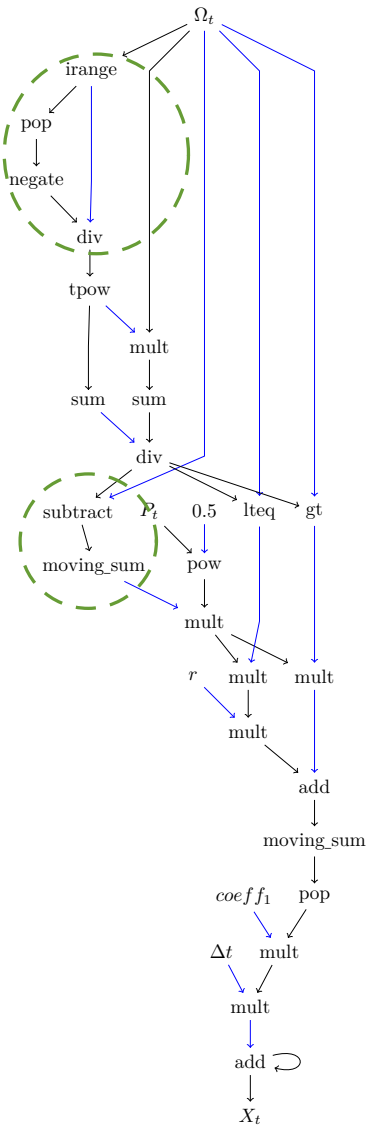


Figure 7: An example evolved version of CGP-ShoreFor.

Considering the results obtained from the graph size experiment as presented in Section 3.3, an interesting path for future work would be the inclusion of additional mutation operators such as node addition and deletion, in order to remove any user-specified limitations and to allow the method to autonomously evolve towards its optimal (active) graph size.

Another possible direction for future work would be to move towards a multi-objective fitness evaluation scheme which we believe would be beneficial on multiple aspects. First, an evaluation procedure that takes into account the complexity and/or modularity of the evolved graph would allow for more direct interpretability of the evolved graphs. Moreover, the inclusion of shoreline datasets from different sites as objectives in the evaluation procedure, in

addition to more specialized tests over specific periods of time (e.g. storms) would allow for more sophisticated experimentation on evolving generalized or specialized models.

Finally, we relied on manual translation of the ShoreFor model into CGP individuals for GI in this work. Analysis on the resulting individuals was done using the graphs shown in this article in comparison to the graph of the original ShoreFor model. Further work on automatically translating mathematical and physical models into CGP individuals for improvement, with post-hoc analysis and translation back into a system of equations also done automatically, would greatly help in the application of GI using CGP to other domains.

6 CONCLUSION

In this work, we have presented our first results in applying CGP in a Genetic Improvement formulation of the problem of forecasting wave-driven shoreline change using the ShoreFor shoreline equilibrium model to initialize our population of CGP individuals. We presented our methodology of transforming the original ShoreFor system of equations into a CGP encoding including the different simplifications that were required to move from a full regression-based model to an instantaneous model that operates on a subset of the full dataset at a time.

We also presented the different experiments that were conducted to improve the efficiency of CGP. We find that using the full calibration series gives a more reliable measure of performance during evolution while increasing the computational overhead. We highlight the possibility of using noisy fitness estimation techniques in order to reduce the computational overhead while keeping a reliable measure of performance during evolution. We discussed the different constraints that we added to the base genetic algorithm and we analyzed their impacts on evolution. We find that applying constraints on the computation graph during the mutation step yields the best performance. We also find that applying constraints based on the predicted time series at the evaluation level also improves convergence. Interestingly, merging both types of constraints did not yield any improvement in performance. The impact of the complete CGP graph size was studied and we found that using smaller graphs usually led to better results, which is a strong motivation for further analysis of additional mutation operators such as node addition and deletion.

Finally, we presented a comparison of performance between the top performing CGP-ShoreFor individuals to pure CGP individuals. The CGP-ShoreFor models were found to improve on the original ShoreFor model on all time scales, while the performance of the pure CGP individuals was superior at very short time scales. Overall, the preliminary results presented in this work are a strong motivation for further studies in using CGP and genetic algorithms in the physical sciences, especially considering the opportunities of physics discovery due to the white-box nature of genetic programming.

REFERENCES

[1] Giuseppe Carleo, Ignacio Cirac, Kyle Cranmer, Laurent Daudet, Maria Schuld, Naftali Tishby, Leslie Vogt-Maranto, and Lenka Zdeborová. 2019. Machine learning and the physical sciences. *Reviews of Modern Physics* 91, 4 (2019), 045002.

[2] Milan Češka, Jiří Matyáš, Vojtech Mrazek, Lukas Sekanina, Zdenek Vasicek, and Tomas Vojnar. 2017. Approximating complex arithmetic circuits with formal error guarantees: 32-bit multipliers accomplished. In *2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 416–423.

[3] John A Church and Neil J White. 2006. A 20th century acceleration in global sea-level rise. *Geophysical research letters* 33, 1 (2006).

[4] MA Davidson, KD Splinter, and IL Turner. 2013. A simple equilibrium model for predicting shoreline change. *Coastal Engineering* 73 (2013), 191–202.

[5] Gregory Duveiller, Dominique Fasbender, and Michele Meroni. 2016. Revisiting the concept of a symmetric index of agreement for continuous datasets. *Scientific reports* 6, 1 (2016), 1–14.

[6] GS Dwarakish, Rakshith Shetty, and Usha Natesan. 2013. Review on applications of neural network in coastal engineering. *Artificial Intelligent Systems and Machine Learning* 5, 7 (2013), 324–331.

[7] Brian W Goldman and William F Punch. 2013. Reducing wasted evaluations in cartesian genetic programming. In *European Conference on Genetic Programming*. Springer, 61–72.

[8] Evan B Goldstein, Giovanni Coco, and Nathaniel G Plant. 2019. A review of machine learning applications to coastal sediment transport and morphodynamics. *Earth-science reviews* 194 (2019), 97–108.

[9] Simon Harding, Jürgen Leitner, and Juergen Schmidhuber. 2013. Cartesian genetic programming for image processing. In *Genetic programming theory and practice X*. Springer, 31–44.

[10] Gonéri Le Cozannet, Thomas Bulteau, Bruno Castelle, Roshanka Ranasinghe, Guy Wöppelmann, Jeremy Rohmer, Nicolas Bernon, Déborah Idier, Jessie Louisor, and David Salas-y Mélia. 2019. Quantifying uncertainties of sandy shoreline change projections as sea level rises. *Scientific reports* 9, 1 (2019), 1–11.

[11] Julian F Miller. 2011. Cartesian genetic programming. In *Cartesian Genetic Programming*. Springer, 17–34.

[12] Julian Francis Miller. 2020. Cartesian genetic programming: its status and future. *Genetic Programming and Evolvable Machines* 21, 1 (2020), 129–168.

[13] Jennifer Montaño, Giovanni Coco, Jose AA Antolínez, Tomas Beuzen, Karin R Bryan, Laura Cagigal, Bruno Castelle, Mark A Davidson, Evan B Goldstein, Raimundo Ibaceta, et al. 2020. Blind testing of shoreline evolution models. *Scientific reports* 10, 1 (2020), 1–10.

[14] Robert J Nicholls, Susan E Hanson, Jason A Lowe, Richard A Warrick, Xianfu Lu, and Antony J Long. 2014. Sea-level scenarios for evaluating coastal impacts. *Wiley Interdisciplinary Reviews: Climate Change* 5, 1 (2014), 129–150.

[15] Magnus Rattray and Jon Shapiro. 1998. Noisy fitness evaluation in genetic algorithms and the dynamics of learning. (1998).

[16] Borja G Reguero, Iñigo J Losada, and Fernando J Méndez. 2019. A recent increase in global wave power as a consequence of oceanic warming. *Nature communications* 10, 1 (2019), 1–14.

[17] Yasuhito Sano and Hajime Kita. 2002. Optimization of noisy fitness functions by means of genetic algorithms using history of search with test of estimation. In *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No. 02Th8600)*, Vol. 1. IEEE, 360–365.

[18] Rob Schepper, Rafael Almar, Erwin Bergsma, Sierd de Vries, Ad Reniers, Mark Davidson, and Kristen Splinter. 2021. Modelling cross-shore shoreline change on multiple timescales and their interactions. *Journal of Marine Science and Engineering* 9, 6 (2021), 582.

[19] Lukas Sekanina, Simon L Harding, Wolfgang Banzhaf, and Taras Kowaliw. 2011. Image processing and CGP. In *Cartesian genetic programming*. Springer, 181–215.

[20] Kristen D Splinter, Ian L Turner, Mark A Davidson, Patrick Barnard, Bruno Castelle, and Joan Oltman-Shay. 2014. A generalized equilibrium model for predicting daily to interannual shoreline response. *Journal of Geophysical Research: Earth Surface* 119, 9 (2014), 1936–1958.

[21] Alexandra Toimil, Inigo J Losada, Paula Camus, and Pedro Díaz-Simal. 2017. Managing coastal erosion under climate change at the regional scale. *Coastal Engineering* 128 (2017), 106–122.

[22] Yen Hai Tran and Eric Barthélemy. 2020. Combined longshore and cross-shore shoreline model for closed embayed beaches. *Coastal Engineering* 158 (2020), 103692.

[23] Yen Hai Tran, Patrick Marchesiello, Rafael Almar, Duc Tuan Ho, Thong Nguyen, Duong Hai Thuan, and Eric Barthélemy. 2021. Combined longshore and cross-shore modeling for low-energy embayed sandy beaches. *Journal of Marine Science and Engineering* 9, 9 (2021), 979.

[24] Andrew James Turner and Julian Francis Miller. 2014. Recurrent cartesian genetic programming. In *International Conference on Parallel Problem Solving from Nature*. Springer, 476–486.

[25] Andrew James Turner and Julian Francis Miller. 2014. Recurrent Cartesian genetic programming applied to famous mathematical sequences. In *Proceedings of the Seventh York Doctoral Symposium on Computer Science & Electronics*. 37–46.

[26] Dennis G Wilson, Sylvain Cussat-Blanc, Hervé Luga, and Julian F Miller. 2018. Evolving simple programs for playing Atari games. In *Proceedings of the Genetic and Evolutionary Computation Conference*. 229–236.

[27] Lynn D Wright, Andrew D Short, and MO Green. 1985. Short-term changes in the morphodynamic states of beaches and surf zones: an empirical predictive model. *Marine geology* 62, 3-4 (1985), 339–364.