Hindawi
Discrete Dynamics in Nature and Society
Volume 2018, Article ID 6848745, 15 pages
https://doi.org/10.1155/2018/6848745

Research Article

Prediction of Drifter Trajectory Using Evolutionary Computation

Yong-Wook Nam and Yong-Hyuk Kim

Department of Computer Science, Kwangwoon University, 20 Kwangwoon-ro, Nowon-gu, Seoul 01897, Republic of Korea

Correspondence should be addressed to Yong-Hyuk Kim; yhdfly@kw.ac.kr

Received 11 September 2017; Revised 6 November 2017; Accepted 19 December 2017; Published 24 January 2018

Academic Editor: Alicia Cordero

We used evolutionary computation to predict the trajectory of surface drifters. The data used to create the predictive model comprise the hourly position of the drifters, the flow and wind velocity at the location, and the location predicted by the MOHID model. In contrast to existing numerical models that use the Lagrangian method, we used an optimization algorithm to predict the trajectory. As the evaluation measure, a method that gives a better score as the Mean Absolute Error (MAE) when the difference between the predicted position in time and the actual position is lower and the Normalized Cumulative Lagrangian Separation (NCLS), which is widely used as a trajectory evaluation method of drifters, were used. The evolutionary methods Differential Evolution (DE), Particle Swarm Optimization (PSO), Covariance Matrix Adaptation Evolution Strategy (CMA-ES), and ensembles of the above were used, with the DE&PSO ensemble found to be the best prediction model. Considering our objective to find a parameter that minimizes the fitness function to identify the average of the difference between the predictive change and the actual change, this model yielded better results than the existing numerical model in three of the four cases used for the test data, at an average of 19.36% for MAE and 5.96% for NCLS. Thus, the model using the fitness function set in this study showed improved results in NCLS and thus shows that NCLS can be used sufficiently in the evaluation system.

1. Introduction

The technology for predicting particle trajectories in the ocean can be used in a variety of ways. For example, it can provide a method to track objects in the ocean during a distress situation or an accident through the last observed time and location data, as well as predicting the path of icebergs floating at the sea. It also presents the possibility of tracing pollutants in the event of accidents such as the 2010 Deepwater Horizon oil spill in the Gulf of Mexico; as a result, numerous studies have been conducted on the matter [1, 2]. Conventionally, a specific equation is used to predict the movement of an object, and the constant parameters used are based on previously studied values. In this study, we set this equation in a form suitable for parameter optimization irrespective of fluid dynamics and predicted the particle trajectory by setting the constant parameter used here to the optimal value through evolutionary computation. This is a novel prediction method, and it is significant in that it suggests a new method of designing a prediction model.

To predict particle trajectories in the ocean, we used several drifters such as those shown in Figure 1 [3, 4]. Drifters observe the weather and ocean conditions while traveling at the sea. Although their capabilities differ from device to device, in general, drifters can observe elements such as flow velocity, wind speeds, temperature, air pressure, and salinity around their position [5]. However, the drifter observing the trajectory usually uses the ability to transmit its position [6]. If we create a model that predicts the trajectory of an object based on all these observation factors, it is difficult to apply it to other similar problems. When a ship accident occurs, the ship's objects usually do not have a sensor that can measure the observed elements, and even if they do, it lacks the ability to transmit such information. Therefore, it is necessary to utilize meteorological elements that can be measured or predicted globally, rather than accurate

Figure 1: Surface drifters [9].

observations surrounding the object. In the end, a drifter merely needs to transmit its exact position, and, thus, it is better to minimize the functionality and weight of the device such that it drifts well in the ocean. In recent years, services have emerged through the development of technology that provides weather and ocean information in a specific area, and such information can be used to visualize the earth's weather information [7]. Furthermore, because such services are usually provided in application programming interface (API) format [8], they can be easily utilized anywhere. This may also be provided to meteorological agencies and may be more reliable than global APIs because the data characteristics are localized to the country of origin. On this basis, this study was conducted with the objective of creating a model that predicts drifter trajectory using real-time meteorological and oceanographic information. This paper makes the following three contributions. First, to the best of our knowledge, this is the first attempt to predict particle trajectories in the ocean through parameter optimization using only evolutionary computation. Second, Normalized Cumulative Lagrangian Separation (NCLS) is widely used for particle trajectory prediction. However, it is not suitable for many fitness functions because it calculates errors cumulatively, to which this paper provides a novel fitness function that can be used to increase the NCLS score even when the computation volume is low. Finally, as a result of the test, in three cases out of four test cases, our method showed better performance than the existing numerical model to be introduced in Section 3.1. On average, the trajectory prediction was improved 19.36% and 5.96% in relation to Mean Absolute Error (MAE) and NCLS, respectively.

The remainder of this paper is organized as follows. Section 2 describes the data and numerical model. Section 3 explains the evolutionary methods and fitness functions used in this study and also describes the evaluation measures of the prediction model. Section 4 describes the experiments conducted and the environments. Section 5 presents concluding remarks.

2. Drifter Data

To design the prediction model in this study, seven drifters were dispatched at different locations from November 6 to October 16, 2015, near the offshore of Sosan in Korea. The location of each drifter was recorded in hours from the first drop in the ocean. The period and the number of datasets measured are shown in Table 1. The larger the number of data is, the longer the location is transmitted.

Wind velocity data were obtained from the Korea Meteorological Administration (KMA). The flow velocity used reanalyzed data provided by ARA Consulting & Technology [6] using factors such as drifter speed, tide, wind, water temperature, and salinity. The actual movement path of the measured drifters is shown in Figure 2.

The training data for predicting drifter movement have the same attributes as those in Table 2.

In other words, the location of the drifters should be determined through the wind and flow velocities. However, because velocity is a variation as well, the value that can be obtained by using the wind velocity and flow velocity should also be the amount of change of the position. As the current data only shows the absolute position of the drifter, the observed location part of the training data should be changed from the absolute position to the positional change amount. Therefore, it can be said that the change of the location as shown in Table 3 indicates that when the wind speed and the flow velocity are somewhat changed, the drifter has moved to an extent.

Observed locations are subtracted from the values in the next line, and the result indicates the changed amount. Therefore, the numerical value obtained by the wind and flow velocities in one row becomes the positional variation and is in a form suitable for training. Unfortunately, data from other studies predicting trajectory [1–4] do not contain the information of wind and flow velocity, so the prediction methods used in our experiments cannot be applied to the data. However, our results could be successfully verified through comparison with the existing numeric model.

3. Prediction Methods

3.1. Existing Numerical Model. The conventional prediction model compared with the results of this experiment uses the MOHID (MOdelo HIDrodinâmico) model. MOHID is a model developed in 1985 at the Marine and Environmental Technology Research Center (MARETEC) of the Instituto Superior Técnico (IST) of the University of Lisbon, Portugal [10], and can be applied to 1, 2, and 3 dimensions of the ocean. The movement of the drifters in the model is calculated as in (1) by applying the Lagrangian method [11].

$$\frac{dx_i}{dt} = u_i\left(x_i t\right), \tag{1}$$

where u is the average flow velocity and x is the particle position. Figure 3 shows the location where the drifters were released and the location predicted by the MOHID model.

3.2. Evolutionary Methods. In this study, we formulated an equation using wind and flow velocity to predict the trajectory of drifters. In the equation, the wind and flow velocity into the variables are combined with the constant parameters

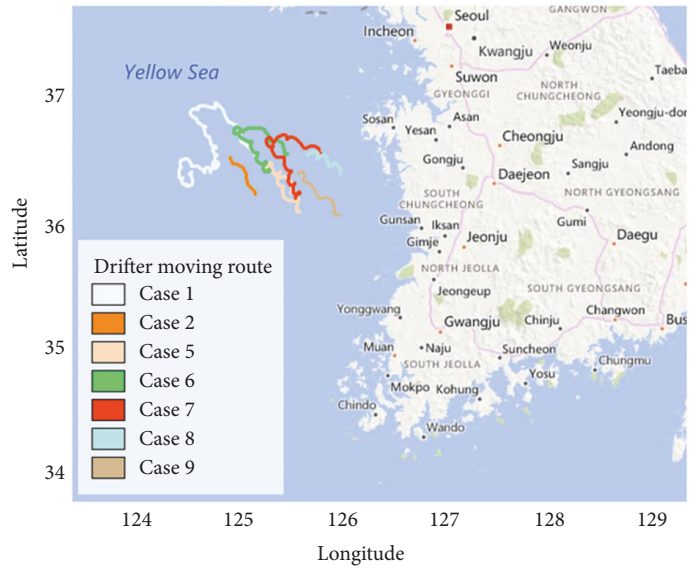


Figure 2: Observed trajectories of the seven drifters used to build our predictive models.

Table 1: Period and number of datasets measured.

	Case 1	Case 2	Case 5	Case 6	Case 7	Case 8	Case 9
Period	9 d 23 h	1 d 2 h	4 d 13 h	4 d 16 h	4 d 17 h	1 d 15 h	1 d 21 h
Number of data	239	26	109	112	113	39	45

Table 2: Attributes of the training data (examples).

Time				Observed location		Wind		Flow	
Year	Month	Day	Hour	Latitude	Longitude	U	V	U	V
2015	11	6	8	125.0796	36.5788	0.1770	−0.0933	−6.2055	−0.1881
2015	11	6	9	125.0731	36.5750	0.1339	−0.0046	−5.6724	0.5072
2015	11	6	10	125.0690	36.5750	0.0909	0.0841	−5.1393	1.2025
2015	11	6	11	125.0633	36.5816	0.0481	0.1726	−4.6124	1.8818

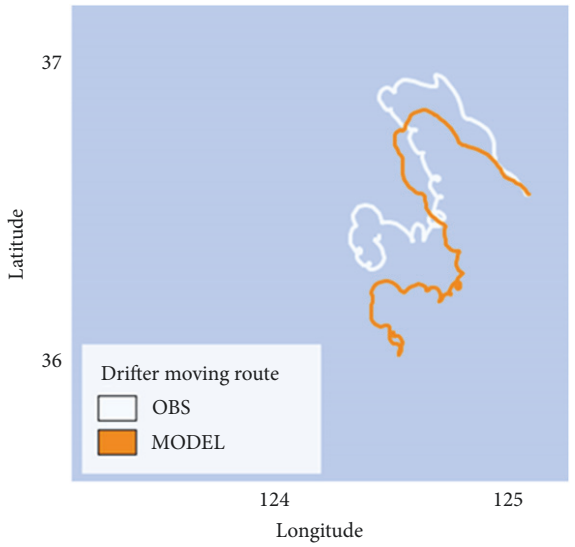


Figure 3: Observed trajectory of drifter (OBS) and predicted one of MOHID model to be compared with our predicted models (Case 1).

to calculate the position variation. Therefore, we need to use an evolutionary method to deal with the real parameter optimization problem. In this study, we used Differential Evolution (DE), Particle Swarm Optimization (PSO), and Covariance Matrix Adaptation Evolution Strategy (CMA-ES), which have had success resolving such problems in the past.

3.2.1. Differential Evolution. Price et al. [12] proposed DE to solve problems using vector differences. DE is a type of evolutionary computation that operates similarly to a genetic algorithm. This algorithm has the advantage that it can be manipulated with a small number of variables to optimize the fitness function. As generations evolve to change generations, the parameters that make up an entity are optimized for the fitness function. Because the type of entity is real number encoding, the above study is in good agreement with the present study for parameter optimization.

3.2.2. Particle Swarm Optimization. Kennedy and Eberhart [13] proposed PSO, which mimics the movement of

Table 3: Attributes of the transformed training data (examples).

Time				Observed location		Wind		Flow	
Year	Month	Day	Hour	Latitude	Longitude	U	V	U	V
2015	11	6	9	−0.0065	−0.0039	0.1770	−0.0933	−6.2055	−0.1881
2015	11	6	10	−0.0041	0.0000	0.1339	−0.0046	−5.6724	0.5072
2015	11	6	11	−0.0057	0.0066	0.0909	0.0841	−5.1393	1.2025
2015	11	6	12	−0.0083	0.0106	0.0481	0.1726	−4.6124	1.8818

Table 4: Number of tuples on the training data.

Data	Case 1	Case 2	Case 5	Case 6	Case 7	Case 8	Case 9
The number of tuples	238	25	108	111	112	38	44

organisms in a bird flock or fish school to solve optimization problems. It is a typical evolutionary computation optimization technique and has excellent execution speed and performance. This algorithm operates on the assumption that individuals belonging to a cluster share information with each other in the course of movement and that an individual belonging to a cluster acts on the basis of information shared throughout the herd, as well as its experience. Each object moves to the optimal position, being influenced by the best location that it has found so far and the best location that neighboring particles have found. Because many objects are searching, it is possible to overcome the disadvantage of convergence to local optima as in, for example, simulated annealing [14].

3.2.3. CMA-ES. Proposed by Hansen et al. [15], CMA-ES is an evolutionary computation technique suitable for solving difficult nonlinear, nonconvex, and black-box optimization problems in continuous domains. This method uses the covariance matrix when distributing objects, enabling better locating of points of interest in relation to the global optima. When individuals reach global optima, the covariance matrix decreases and eventually converges. It has high performance in solving problems by using evolutionary computation, and its parameters need to be set better than other optimization techniques [16].

3.2.4. Ensemble. An ensemble is a technique that generates multiple models and combines the predicted results of each model to generate new results [17]. It is used to obtain a predicted value with a degree of higher reliability compared to the results of a single classifier by combining the prediction results of several classifiers via machine learning. In this study, we used four different types designed by combining algorithms from Sections 3.2.1 to 3.2.3 (i.e., DE&PSO, PSO&CMA-ES, DE&CMA-ES, and DE&PSO&CMA-ES). The numerical value predicted by each algorithm is a variation. The ensemble model allows the combination algorithms to calculate the average of the predicted changes over time and to have the characteristics of each algorithm.

The above algorithms are suitable for real number parameter optimization problems and can all have a fitness function. The attributes that can be used as parameters in Table 3 are u and v of wind and flow velocity, respectively, and the resulting value is the latitude and longitude of the observed location. Therefore, in this experiment, the fitness function was set as shown in (2), with parameter being obtained set as a_i.

$$\begin{aligned}\frac{1}{n}\sum_{i=1}^{n}\big(&\big|a_1\text{velo\_}u_i + a_2\text{velo\_}v_i + a_3\text{velo\_}u_i * \text{velo\_}v_i \\ &+ a_4\text{wind\_}u_i + a_5\text{wind\_}v_i + a_6\text{wind\_}u_i * \text{wind\_}v_i \\ &+ a_7 - \text{observed\_Lat}\big| + \big|a_8\text{velo\_}u_i + a_9\text{velo\_}v_i \\ &+ a_{10}\text{velo\_}u_i * \text{velo\_}v_i + a_{11}\text{wind\_}u_i + a_{12}\text{wind\_}v_i \\ &+ a_{13}\text{wind\_}u_i * \text{wind\_}v_i + a_{14} - \text{observed\_Lon}\big|\big),\end{aligned} \tag{2}$$

where velo_u and velo_v are the flow velocity of the training data and wind_u and wind_v are the wind speeds of the training data, respectively. observed_Lat and observed_Lon are the change rate of the training data, which is the number to predict using the flow velocity and wind speed, and n is the number of training datasets. In evolutionary computation, the value to be minimized is the sum of the differences in latitude and longitude change rates of the predicted point and the actual point. The aim of the experiment is to find a parameter a_i that has as few errors as possible when a model is created and the test data are included in the above equation.

4. Experiments

4.1. Setting and Environments. All experiments were tested through cross validation. Cross validation is a process in which all of the data are divided into N number of subsamples, learning $N-1$ subsamples, and testing the remaining one through N times [18]. However, in this experiment, because seven regions of data are different from each other, it is not suitable to go through the process of dividing all the gathered data. Therefore, it is best to test with all the data excluding the data to be tested and then perform verification with the test data. Table 4 shows the number of tuples in the transformed data.

Cases 2, 8, and 9 were used only as training data and were excluded from the test because the number of tuples was too small. After calculating the optimal parameters for test data with each evolutionary method, we summarized all

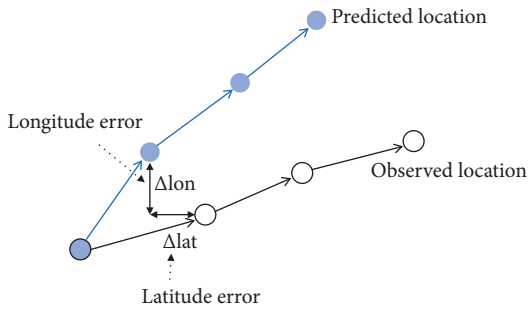


Figure 4: Longitude error and latitude error of predicted location.

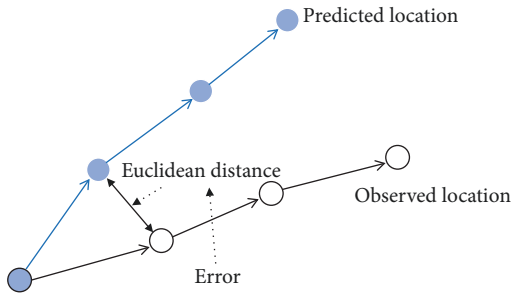


Figure 5: Error defined by Euclidean distance between the predicted location and the observed one.

predicted values. The predicted value is the amount of change in position, such as transformed data. To verify this, all must be reverted to the region. Therefore, it keeps changing the position based on the first area of the original data that is not transformed. As a result, the time-based prediction area remains, and all the measures discussed in Section 3 can be verified.

Each evolutionary method can set parameters to suit the experimental situation. In this experiment, the most suitable parameters were found and the best method was found by comparing the results representing each evolutionary method. The best fitness and average fitness of the population according to each generation of the evolutionary method are also displayed in a graph; thus, the performance is shown graphically. The computer used in this experiment had an Intel i7 7700k (4.2 GHz) CPU and the evolutionary method was implemented in the C language.

4.2. Evaluation Measures

4.2.1. Mean Absolute Error (MAE). As shown in Figure 4, the latitude and longitude differences between the predicted location and the observed location over time are set to error as a simple way to verify from the given data divided by latitude and longitude. After this, the error at all points of the test data is calculated and an average is also calculated.

This can be expressed as in

$$\frac{1}{m}\sum_{i=1}^{m}\left(\left|\text{pred\_Lat}_i - \text{observed\_Lat}_i\right| + \left|\text{pred\_Lon}_i - \text{observed\_Lon}_i\right|\right), \tag{3}$$

where pred_Lon represents the longitude of the predicted data and pred_Lat represents the latitude of the predicted data. observed_Lon shows the longitude of the observed data and observed_Lat shows the latitude of the observed data, and each difference is set as an error. m is the number of test datasets and is divided to obtain the average of the error.

4.2.2. Euclidean Distance. The principle of operation of the algorithm is almost the same as that of Section 4.1, but the error is based on the Euclidean distance rather than the difference between latitude and longitude, as shown in Figure 5.

This can be expressed by

$$\frac{1}{m}\sum_{i=1}^{m}\left(\sqrt{\left(\text{pred\_Lat}_i - \text{observed\_Lat}_i\right)^2 + \left(\text{pred\_Lon}_i - \text{observed\_Lon}_i\right)^2}\right). \tag{4}$$

The description of the variables used is the same as in (3).

4.2.3. Normalized Cumulative Lagrangian Separation (NCLS). NCLS, also called skill score, was developed by Liu and Weisberg [19]. It is widely used as an evaluation method of trajectory modeling [20, 21] and was proposed to solve weaknesses in the Lagrangian separation distance in relation to the continental shelf and its adjacent deep ocean. The evaluation methods in Sections 4.1 and 4.2 are the average of the errors, and thus the lower the better. However, the skill score is found by subtracting the error from one, as shown in Figure 6. Therefore, the higher the value is, the better it is; the maximum value is one.

The tolerance threshold is a number that prevents the skill score from being low when s is too large, depending on the size of the data. Usually, it does not matter if it is set to one (it is also set to one in this study). This method is currently the most widely used trajectory evaluation method. However, as the equation for obtaining s is computationally expensive owing to its use of the cumulative sum, and because the prediction result point is used, the fitness function cannot be left as such. Therefore, the fitness function is defined as the difference between the predictive change and the actual change as shown in (2). Next, the fitness function is used to construct a trajectory model using the NCLS as the evaluation model, and its validity is verified.

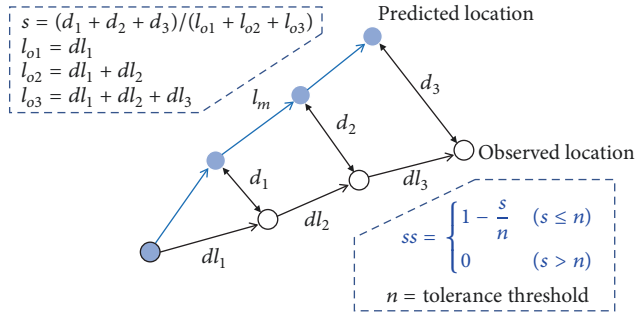


Figure 6: Formula for calculating skill score (ss) of NCLS.

Table 5: Strategies of DE.

Exponential	DE/best/1
	DE/rand/1
	DE/rand-to-best/1
	DE/rand/2
	DE/best/2
Binomial	DE/best/1
	DE/rand/1
	DE/rand-to-best/1
	DE/rand/2
	DE/best/2

4.3. Results

4.3.1. Differential Evolution. To utilize DE, a program distributed by Storn [22] was used. The main parameters of DE are strategies. Table 5 shows the types of strategies available.

As shown in the table, "Exponential" and "Binomial" can be selected. As there is no difference between the two methods, "Exponential" was tested; the results obtained are shown in Table 6.

Each value represents the evaluation value of the best individual prediction among the population of 20,000 generations. "MAE" means the average error value from (3) as described in Section 4, and "Euclid" means the average distance difference from (4). Therefore, the lower the two values, the better. However, since the skill score of "NCLS" is a structure that subtracts the error from the maximum value 1, the higher the better. Positive results are expressed in italic. There is no difference between "DE/rand/1" and "DE/rand-to-best/1." In all cases except Case 1, "DE/rand/1" and "DE/rand-to-best/1" showed the best performances, and thus the numerical value of "DE/rand/1" was used as a representative result of DE. In evolutionary computation, how quickly the population converges and the average performance of the population are also important. Figure 7 shows error values of the best individual and the average error of the population according to generations by performing each case. Therefore, DE can be considered to sufficiently search the parameter space.

The average fitness and best fitness of each population were not significantly different, and it was found that they converged relatively quickly. Table 7 shows the execution time of the training data for each case. The learning time was overall proportional to the number of learning samples.

4.3.2. Particle Swarm Optimization. A program distributed by Kyriakos was used for PSO [23]. There is Inertia in the parameter, and because it is real values type, 0.3, 0.5, and 0.9 were arbitrarily added. The experimental results are shown in Table 8.

The result for the parameter was not significantly different. The larger the Inertia is, the worse the experimental results were. As the Inertia, we chose the value 0.3 showing the best performance. Figure 8 also shows that the PSO is executed for each case. The error value according to the generation is graphically displayed.

The initial error was very high compared to DE, but, after 100 generations, it was close to zero. Therefore, PSO can also search the parameter space sufficiently. The execution speed was faster than DE. Table 9 shows the computing time for making training data for each case with the PSO algorithm. The learning time of PSO was faster than that of DE.

4.3.3. CMA-ES. In the CMA-ES experiment, a source code developed directly by Hansen was used [24]. Although this algorithm also has some parameters that can be modified, the user does not have to modify its values. However, the parameters that can be modified were slightly modified to give the results shown in Table 10.

The default values (weight = log and $\lambda = 100$) were used as there is very little difference according to the parameters. Figure 9 shows the visual representation of the error values according to generations, obtained by performing CMA-ES for each case. DE and PSO continue to run for up to 20,000 cycles even after the optimization is completed. However, CMA-ES did not require a large number of cycles compared to other evolutionary algorithms in the convergence. Until the convergence, large fluctuation was shown.

The lower the iteration, the faster the execution. Table 11 shows the computing time taken to make the training data for each case with the CMA-ES algorithm. In learning, CMA-ES was the fastest among the three methods.

4.3.4. Ensemble and Integration. The ensemble was assembled with the four cases using DE, PSO, and CMA-ES, as

Table 6: Results on parameters of DE.

Parameter	Evaluation	Case			
		1	5	6	7
Strategy					
DE/best/1	MAE	0.1351	0.0761	0.0494	0.1308
	Euclid	0.1772	0.1218	0.0840	0.1931
	NCLS	0.8879	0.8153	0.8895	0.7627
DE/rand/1	MAE	0.0920	*0.0828*	*0.0392*	*0.0653*
	Euclid	0.15402	*0.1342*	*0.0593*	*0.0956*
	NCLS	0.9025	*0.7965*	*0.9220*	*0.8826*
DE/rand-to-best/1	MAE	0.0920	*0.0828*	*0.0392*	*0.0653*
	Euclid	0.1540	*0.1342*	*0.0593*	*0.0956*
	NCLS	0.9025	*0.7965*	*0.9220*	*0.8826*
DE/best/2	MAE	*0.0889*	0.0842	0.0825	0.0966
	Euclid	*0.1337*	0.1364	0.1259	0.1400
	NCLS	*0.9154*	0.7854	0.8344	0.8280
DE/rand/2	MAE	17.0368	9.6299	19.0175	2.8862
	Euclid	27.0270	14.6791	28.9367	4.2267
	NCLS	−16.1006	−21.0442	−37.1143	−4.1928

Note. The lower the MAE and Euclidean values, the better. The higher the NCLS values, the better.

Table 7: Computing time of DE.

Data	Case 1	Case 5	Case 6	Case 7
Computing time (CPU second)	19.4	23.8	23.9	24.3

Table 8: Results on parameters of PSO.

Parameter	Evaluation	Case			
		1	5	6	7
Inertia					
0.3	MAE	*0.0907*	*0.0820*	*0.0385*	*0.0622*
	Euclid	*0.1512*	*0.1330*	*0.0584*	*0.0913*
	NCLS	*0.9044*	*0.7984*	*0.9232*	*0.8878*
0.5	MAE	0.0914	0.0822	0.0386	0.0643
	Euclid	0.1529	0.1331	0.0586	0.0942
	NCLS	0.9033	0.7982	0.9230	0.8843
0.7298 (default)	MAE	0.0916	0.0822	0.0386	0.0643
	Euclid	0.1531	0.1332	0.0585	0.0942
	NCLS	0.9032	0.7981	0.9230	0.8843
0.9	MAE	0.0917	0.0823	0.0385	0.0643
	Euclid	0.1534	0.1332	0.0583	0.0943
	NCLS	0.9030	0.7980	0.9233	0.8842

Note. The lower the MAE and Euclidean values, the better. The higher the NCLS values, the better.

described in Section 3.2.4, and the performance of the model included in the initial data was also measured on the same basis. The results are shown in Table 9.

The data can be visualized as shown in Figure 10. For convenient comparison with the existing numerical model (MOHID model), a dotted line is marked based on the score of the numerical model. In Cases 1, 5, and 6, all the evolutionary methods were superior to the numerical model, but, in Case 7, they did not.

Compared with the existing MOHID model, the performance improvement can be examined through

$$100 \times \left(\frac{\text{Model\_err} - \text{Pred\_err}}{\text{Model\_err}} \right) (\%), \quad (5)$$

where Model_err is the average error of the MOHID model and Pred_err is the average error of the evolutionary methods. If the result is a negative, it signifies that the performance is worse than the MOHID Model. Tables 13 and 14 show

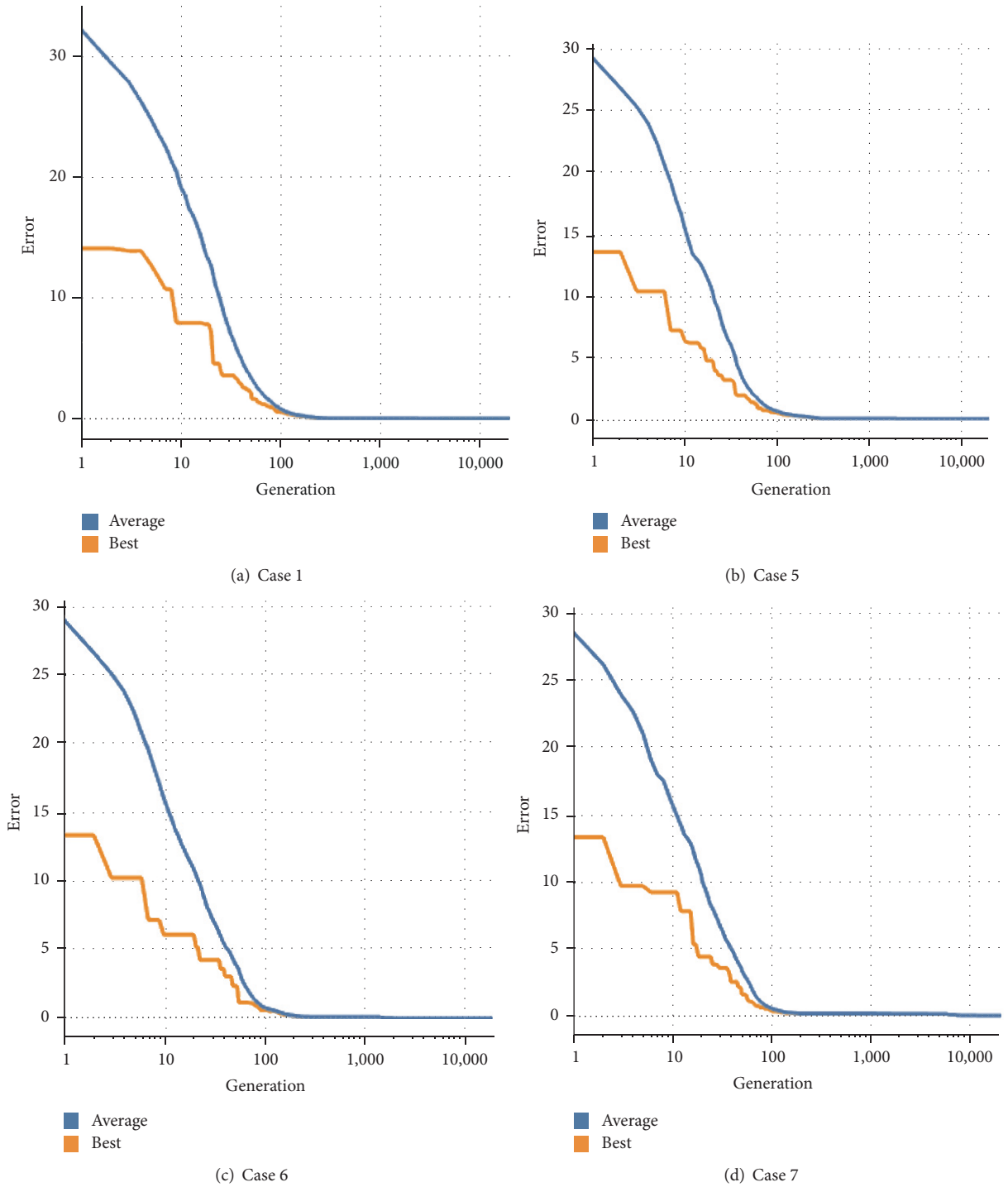


Figure 7: Line graph of showing prediction errors over the number of generations in DE on four test cases.

Table 9: Computing time of PSO.

Data	Case 1	Case 5	Case 6	Case 7
Computing time (CPU second)	5.3	6.6	6.7	7.0

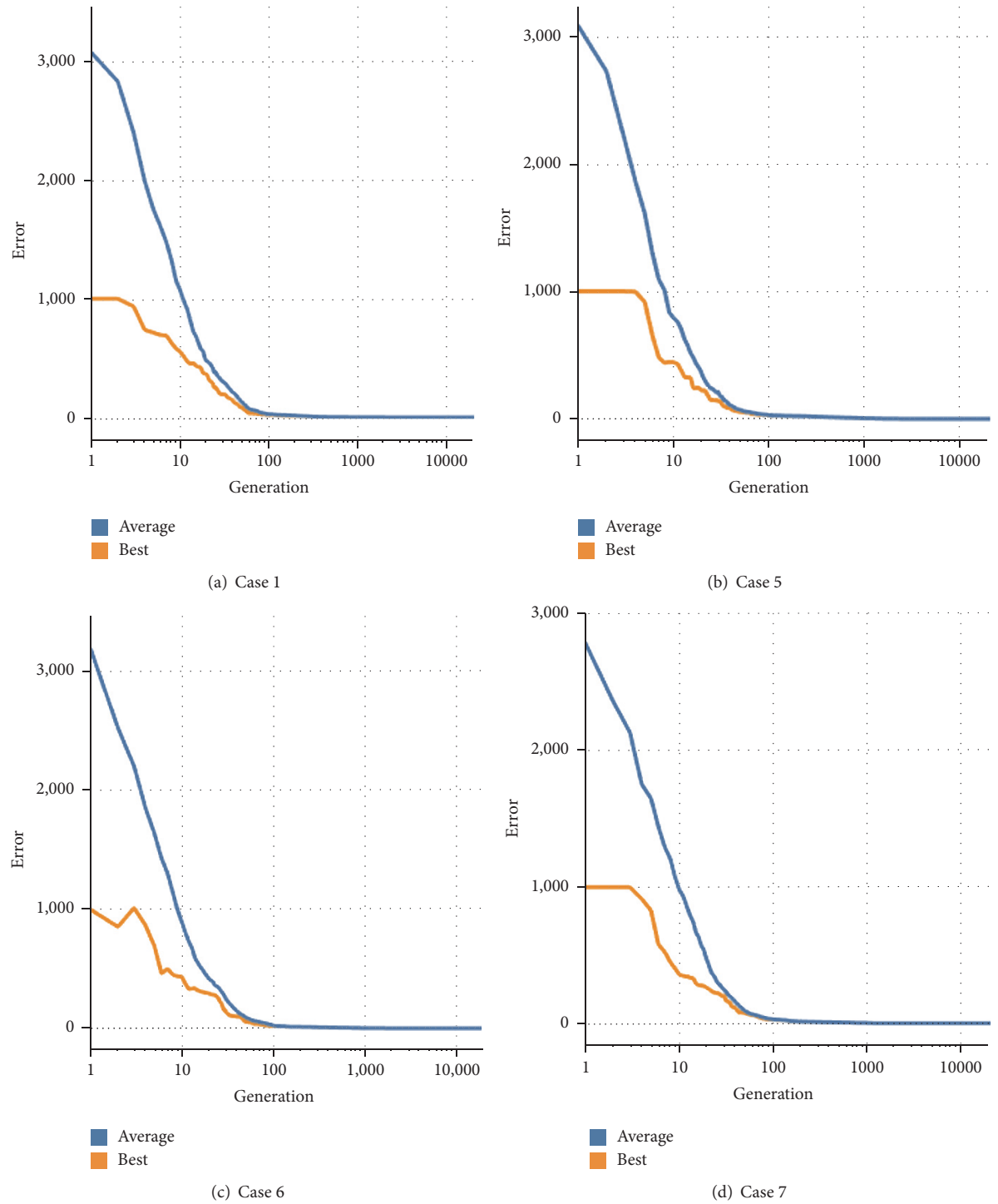


Figure 8: Line graph of showing prediction errors over the number of generations in PSO on four test cases.

the performance improvement values of the evolutionary methods covered in this experiment.

The MAE values showed the best performance with Ensemble (DE&PSO), by 19.36% compared to MOHID. Euclidean values also showed an improvement via Ensemble (DE&PSO) by 18.71% compared to MOHID. Next, the larger the NCLS, called the skill score, the better. The results are shown in Figure 11 as a graph. The results were better than those of the numerical model in Cases 1, 5, and 6, but they were worse in Case 7.

Compared with the existing MOHID model, the performance improvement value can be identified through

$$100 \times \left(\frac{\text{Pred\_err} - \text{Model\_err}}{\text{Model\_err}} \right) (\%) . \tag{6}$$

Table 10: Results on parameters of CMA-ES.

Parameter			Evaluation	Case			
				1	5	6	7
Weight	Log	λ = 100	*MAE*	*0.1303*	*0.1392*	*0.0760*	*0.1305*
			Euclid	*0.2165*	*0.2299*	*0.1188*	*0.2199*
			NCLS	*0.8631*	*0.6513*	*0.8437*	*0.7297*
		λ = 200	MAE	0.1303	0.1393	0.0762	0.1304
			Euclid	0.2165	0.2300	0.1190	0.2199
			NCLS	0.8631	0.6511	0.8438	0.7299
	Linear	λ = 100	MAE	0.1303	0.1393	0.0761	0.1304
			Euclid	0.2165	0.2300	0.1189	0.2200
			NCLS	0.8631	0.6510	0.8438	0.7299
		λ = 200	MAE	0.1303	0.1393	0.0760	0.1304
			Euclid	0.2165	0.2300	0.1188	0.2199
			NCLS	0.8631	0.6510	0.8435	0.7299

Note. The lower the MAE and Euclidean values, the better. The higher the NCLS values, the better.

Table 11: Computing time of CMA-ES.

Data	Case 1	Case 5	Case 6	Case 7
Computing time (CPU second)	3.5	3.2	2.9	2.8

Table 12: Results on parameters of all the tested methods.

Method	Evaluation	Case			
		1	5	6	7
DE, PSO	MAE	0.0900	*0.0796*	0.0334	0.0611
	Euclid	*0.1501*	*0.1295*	0.0514	0.0896
	NCLS	*0.9051*	*0.8035*	0.9324	0.8900
PSO, CMA-ES	MAE	0.0962	0.1051	0.0391	0.0851
	Euclid	0.1656	0.1754	0.0587	0.1431
	NCLS	0.8963	0.7338	0.9228	0.8242
CMA-ES, DE	MAE	0.0962	0.1048	0.0386	0.0867
	Euclid	0.1656	0.1756	0.0580	0.1459
	NCLS	0.8953	0.7334	0.9237	0.8207
DE, PSO CMA-ES	MAE	*0.0885*	0.0943	*0.0268*	0.0751
	Euclid	0.1537	0.1593	*0.0404*	0.1224
	NCLS	0.9028	0.7582	*0.9470*	0.8495
DE (rand/1)	MAE	0.0920	0.0828	0.0392	0.0653
	Euclid	0.1541	0.1342	0.0593	0.0956
	NCLS	0.9026	0.7965	0.9220	0.8826
PSO (Inertia = 0.3)	MAE	0.0907	0.0820	0.0385	0.0622
	Euclid	0.1512	0.1330	0.0584	0.0913
	NCLS	0.9044	0.7984	0.9232	0.8878
CMA-ES (weight = log, λ = 100)	MAE	0.1303	0.1393	0.0761	0.1304
	Euclid	0.2165	0.2300	0.1189	0.2200
	NCLS	0.8631	0.6513	0.8437	0.7297
MOHID model [10]	MAE	0.1352	0.1238	0.0656	*0.0434*
	Euclid	0.2161	0.1890	0.1155	*0.0628*
	NCLS	0.8633	0.7134	0.8480	*0.9229*

Note. The lower the MAE and Euclidean values, the better. The higher the NCLS values, the better.

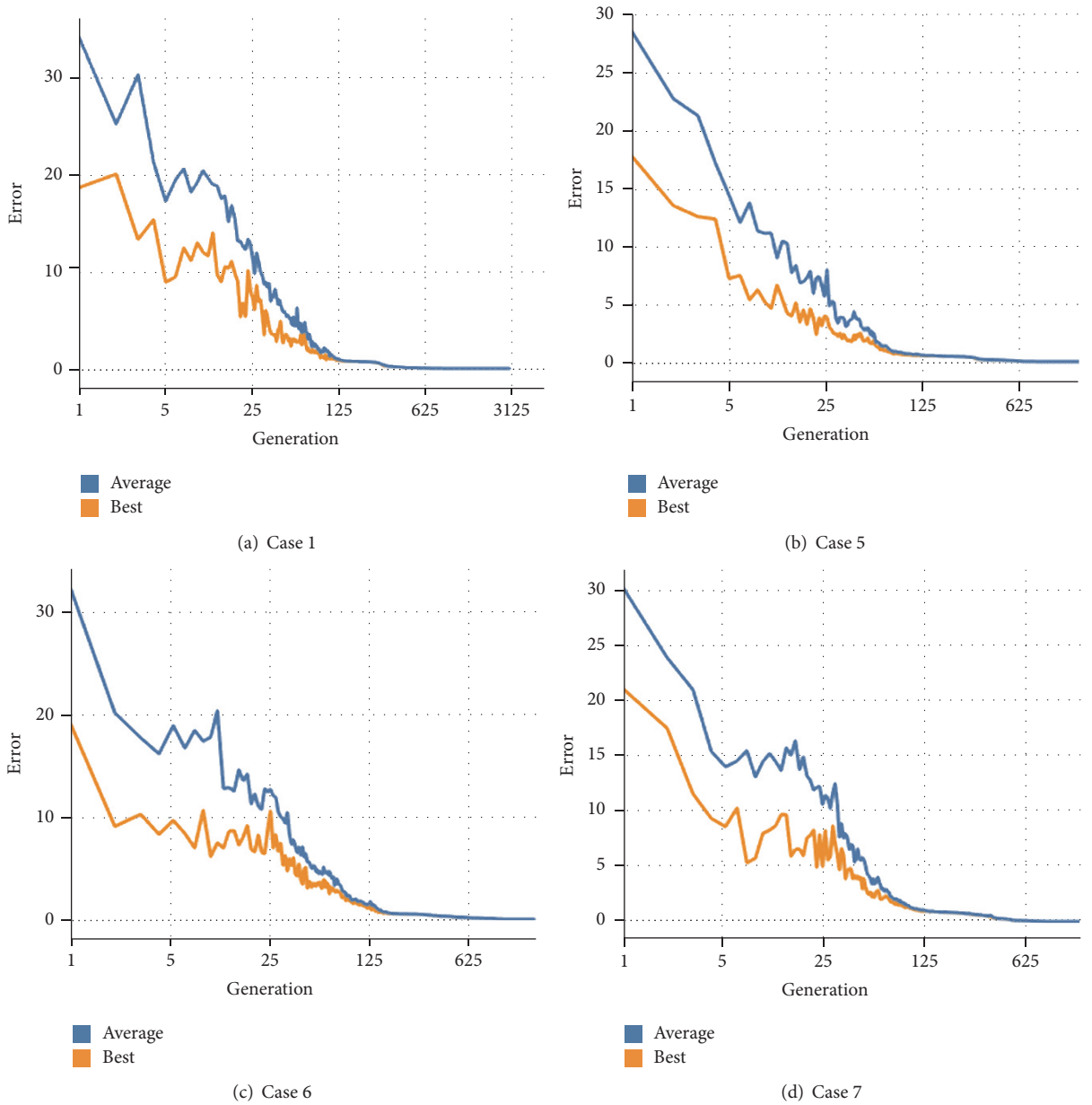
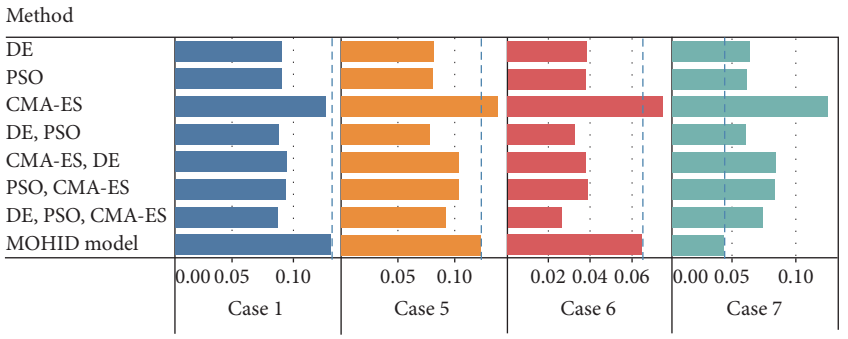


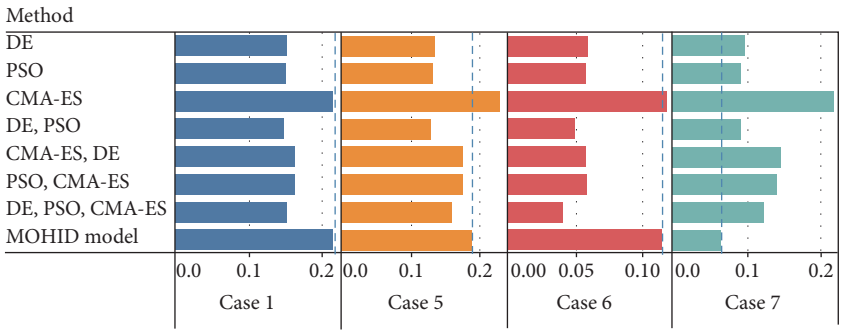
Figure 9: Line graph of showing prediction errors over the number of generations in CMA-ES on four test cases.

Table 13: Performance improvement with respect to MAE (%).

MAE	Case 1	Case 5	Case 6	Case 7	Average
DE&PSO	33.43	*35.70*	49.09	*−40.78*	*19.36*
PSO&CMA-ES	28.85	15.11	40.40	−96.08	−2.93
CMA-ES&DE	28.85	15.35	41.16	−99.77	−3.60
DE&PSO&CMA-ES	*34.54*	23.83	*59.15*	−73.04	11.12
DE	31.95	33.12	40.24	−50.46	13.71
PSO	32.91	33.76	41.31	−43.32	16.17
CMA-ES	3.62	−12.52	−16.01	−200.46	−56.34



(a) MAE



(b) Euclidean distance

Figure 10: Bar graph of MAE and Euclidean distance for all the tested predictive models on four test cases. The lower the values, the better.

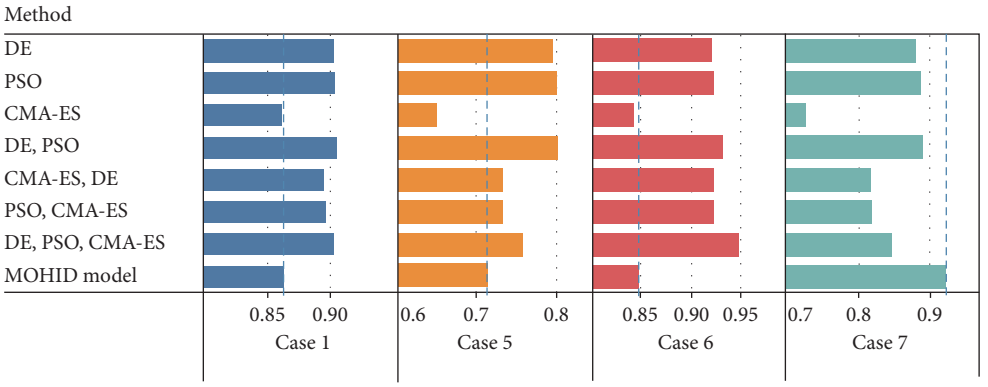


Figure 11: Bar graph of NCLS skill score for all the tested predictive models on four test cases. The higher the values, the better.

Table 14: Performance improvement with respect to Euclidean distance (%).

Euclidean distance	Case 1	Case 5	Case 6	Case 7	Average
DE&PSO	*30.54*	*31.48*	55.50	*−42.68*	*18.71*
PSO&CMA-ES	23.37	7.20	49.18	−127.87	−12.03
CMA-ES&DE	23.37	7.09	49.78	−132.32	−13.02
DE&PSO&CMA-ES	28.88	15.71	*65.02*	−94.90	3.68
DE	28.69	28.99	48.66	−52.23	13.53
PSO	30.03	29.63	49.44	−45.38	15.93
CMA-ES	−0.19	−21.69	−2.94	−250.32	−68.79

Table 15: Performance improvement with respect to NCLS (%).

NCLS	Case 1	Case 5	Case 6	Case 7	Average
DE&PSO	*4.84*	*12.63*	9.95	*−3.56*	*5.96*
PSO&CMA-ES	3.82	2.86	8.82	−10.69	1.20
CMA-ES&DE	3.71	2.80	8.93	−11.07	1.09
DE&PSO&CMA-ES	4.58	6.28	*11.67*	−7.95	3.64
DE	4.55	11.65	8.73	−4.37	5.14
PSO	4.76	11.91	8.87	−3.80	5.44
CMA-ES	−0.02	−8.70	−0.51	−20.93	−7.54

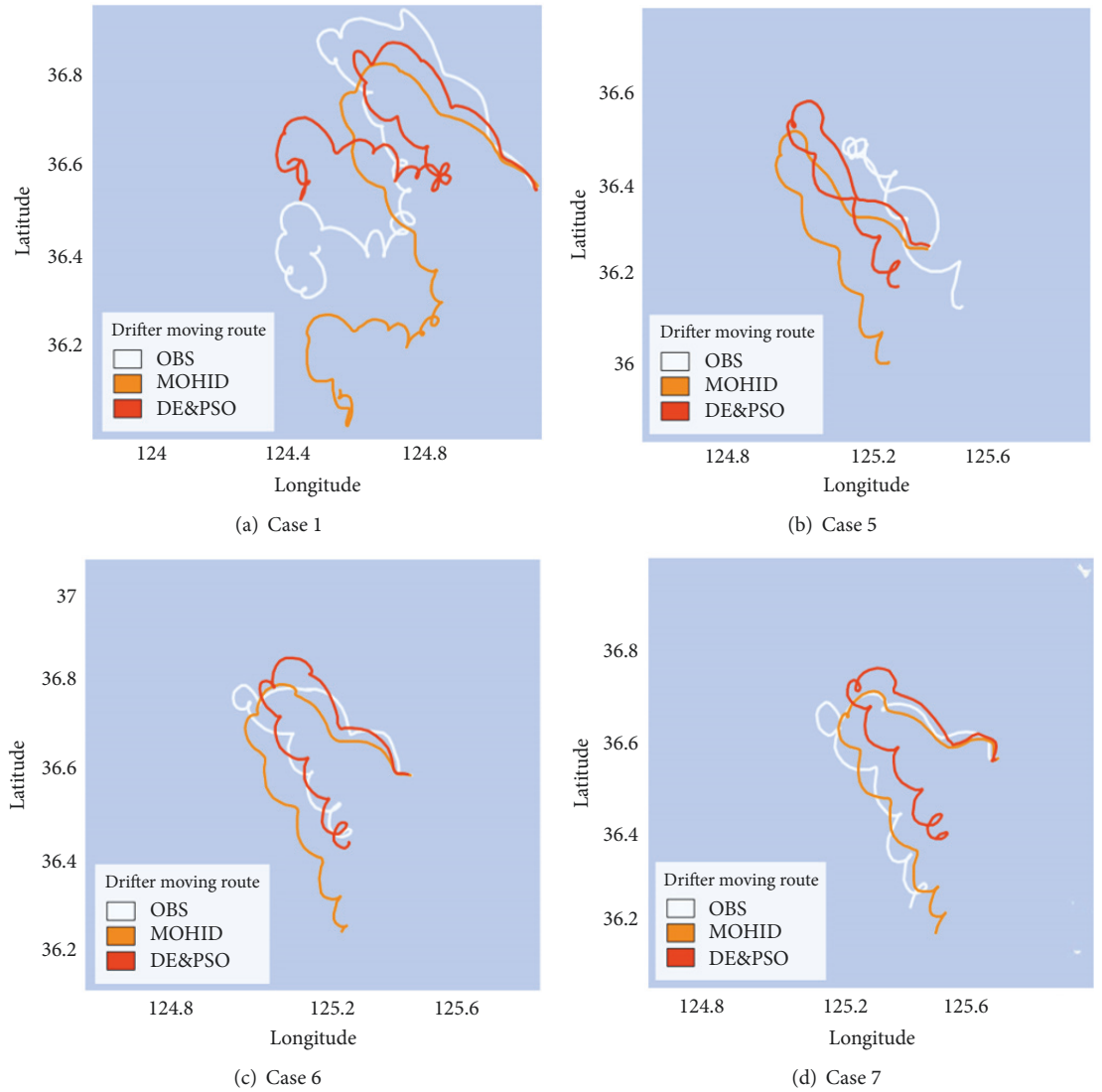


Figure 12: Comparison of predicted trajectory by our ensemble of DE&PSO, one by existing numerical model (MOHID) and observed one for four major drifters.

The definitions of the variables are the same as in (5), but the positions of Pred_err and Model_err have changed because higher values indicate greater improvements. Table 15 shows the performance improvement values of the evolutionary methods based on the NCLS evaluation criteria.

CMA-ES exhibited the worst performances, whereas the Ensemble (DE&PSO), which showed good performance in MAE and Euclid, showed the best performance in NCLS, at 5.96% better than the MOHID model. Tables 10, 11, and 12 all have the same rank regarding average values. This indicates that MAE follows the NCLS standard and shows that (2)

may be used as an evaluation function when constructing a trajectory model based on the NCLS evaluation criterion.

The trajectory (DE&PSO) predicted by the DE&PSO ensemble showing the actual movement path (OBS) in Cases 1, 5, 6, and 7 was selected as the test data. The trajectory predicted by the MOHID model is shown in Figure 12. In Cases 1, 5, and 6, DE&PSO showed better results than the numerical model (MOHID), and in Case 7 better results were shown in the numerical model. Overall, DE&PSO predicted latitude higher than that of observation point and MOHID predicted latitude lower than that of observation point. We guess that this is because the MOHID model is more sensitive to wind speed and flow velocity than the DE&PSO for the predicted latitude.

5. Conclusion

In this paper, we proposed a novel method for predicting drifter trajectory using evolutionary computation. The study is significant in that it is the first to perform parameter optimization using evolutionary computation in predicting particle trajectories in the ocean. In three of the four cases, the trajectory was more accurate than the existing MOHID model. In addition, the fitness function of the evolutionary computation was set as the difference between the observation change rate and the position prediction change rate according to flow velocity and wind speed. The predicted model showed excellent performance of 19.36% on MAE and 5.96% on NCLS on average. Therefore, it is clear that the fitness function can be utilized to increase the NCLS score. In the future, we plan to use machine learning techniques instead of evolutionary methods along with more data. Furthermore, in the current ensemble, all algorithms are combined in an equal ratio. We plan to use the weighted voting ensemble [25] to predict better results. This method can be used to track pollutants in the event of a marine accident. Thus, it is expected that a better trajectory prediction model can be created by combining the existing trajectory prediction technique and the evolutionary computation discussed in this paper.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This research was supported by a grant [KCG-01-2017-05] through the Disaster and Safety Management Institute funded by Korea Coast Guard of Korean government. The authors would like to thank Mr. Do-Youn Kim, a director in ARA Consulting & Technology, for providing the drifter data.

References

[1] T. M. Özgökmen, L. I. Piterbarg, A. J. Mariano, and E. H. Ryan, "Predictability of drifter trajectories in the tropical Pacific Ocean," *Journal of Physical Oceanography*, vol. 31, no. 9, pp. 2691–2720, 2001.

[2] S. Castellari, A. Griffa, T. M. Özgökmen, and P.-M. Poulain, "Prediction of particle trajectories in the adriatic sea using Lagrangian data assimilation," *Journal of Marine Systems*, vol. 29, no. 1-4, pp. 33–50, 2001.

[3] K. R. Thompson, J. Sheng, P. C. Smith, and L. Cong, "Prediction of surface currents and drifter trajectories on the inner Scotian shelf," *Journal of Geophysical Research: Oceans*, vol. 108, no. 9, pp. 3-1, 2003.

[4] C. D. Winant, D. J. Alden, E. P. Dever, K. A. Edwards, and M. C. Hendershott, "Near-surface trajectories off central and southern California," *Journal of Geophysical Research: Oceans*, vol. 104, no. 7, Article ID 1999JC900083, pp. 15713–15726, 1999.

[5] The Global Drifter Program: What's a drifter?, http://www.aoml.noaa.gov/phod/dac/gdp_drifter.php.

[6] ARA Consulting Technology, URL, http://www.aracnt.com/.

[7] Earth:: a global map of wind, weather and ocean condition, https://earth.nullschool.net/.

[8] OpenWeatherMap, https://openweathermap.org/api.

[9] https://www.nefsc.noaa.gov/press_release/2012/SciSpot/SS1211/.

[10] C. M. Allen, "Numerical simulation of contaminant dispersion in estuary flows," *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, vol. 381, pp. 179–194.

[11] C.-Y. Yang, Y.-F. Liaw, C.-M. Chu, and I.-S. Sheen, "White count, pH and lactate in ascites in the diagnosis of spontaneous bacterial peritonitis," *Hepatology*, vol. 5, no. 1, pp. 85–90, 1985.

[12] K. Price, R. M. Storn, and J. A. Lampinen, *Differential Evolution: a Practical Approach to Global Optimization*, Springer Science & Business Media, 2006.

[13] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks*, pp. 1942–1948.

[14] L. Lamberti, "An efficient simulated annealing algorithm for design optimization of truss structures," *Computers & Structures*, vol. 86, no. 19-20, pp. 1936–1953, 2008.

[15] N. Hansen, S. D. Müller, and P. Koumoutsakos, "Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES)," *Evolutionary Computation*, vol. 11, no. 1, pp. 1–18, 2003.

[16] The CMA Evolution Strategy, https://www.lri.fr/~hansen/cmaesintro.html.

[17] Y. Liu, X. Yao, and T. Higuchi, "Evolutionary ensembles with negative correlation learning," *IEEE Transactions on Evolutionary Computation*, vol. 4, no. 4, pp. 380–387, 2000.

[18] P. Refaeilzadeh, L. Tang, and H. Liu, "Cross-validation," in *Encyclopedia of Database Systems*, pp. 532–538, Springer, USA, 2009.

[19] Y. Liu and R. H. Weisberg, "Evaluation of trajectory modeling in different dynamic regions using normalized cumulative Lagrangian separation," *Journal of Geophysical Research: Oceans*, vol. 116, no. 9, Article ID C09013, 2011.

[20] R. Sorgente, C. Tedesco, F. Pessini et al., "Forecast of drifter trajectories using a Rapid Environmental Assessment based on CTD observations," *Deep-Sea Research Part II: Topical Studies in Oceanography*, vol. 133, pp. 39–53, 2016.

[21] Y. Liu, R. H. Weisberg, S. Vignudelli, and G. T. Mitchum, "Evaluation of altimetry-derived surface current products using Lagrangian drifter trajectories in the eastern Gulf of Mexico,"

Journal of Geophysical Research: Oceans, vol. 119, no. 5, pp. 2827–2842, 2014.

[22] Differential Evolution, http://www1.icsi.berkeley.edu/~storn/code.html.

[23] CMA-ES Source Code, https://www.lri.fr/~hansen/cmaes_inmatlab.html.

[24] Particle Swarm Optimization (PSO) in C, https://github.com/kkentzo/pso.

[25] L. I. Kuncheva and J. J. Rodríguez, "A weighted voting framework for classifiers ensembles," *Knowledge and Information Systems*, vol. 38, no. 2, pp. 259–275, 2014.