Dynamical Combinatorial Optimization for Predicting Multivariate Complex Systems

Tomoya Suzuki

Department of Intelligent Systems Engineering, College of Engineering, Ibaraki University 4-12-1 Nakanarusawa-cho, Hitachi, Ibaraki 316-8511, Japan
E-mail: tsuzuki@mx.ibaraki.ac.jp

Abstract Real systems are often composed of many elements interacting with each other and show complex behavior. To predict these complex systems, we can refer to their past behavior, but all of the observed elements do not always compose the same system. Thus, we must detect some essential elements from the observed elements so as to improve the prediction accuracy of learning data. Moreover, if we apply Takens’s embedding theorem to reconstruct an attractor with only a single element, we need not select elements, but we must optimize embedding parameters. In any case, because we must solve the above optimization problems, we applied the genetic algorithm (GA) as an example of metaheuristic techniques. Moreover, real systems might be nonstationary and their own mechanism changes dynamically. Therefore, we reiterated the GA for each optimization with simple algorithms to embed and to predict time-series data to save numerical costs. Through some simulations, we confirmed that our dynamical optimization can improve the prediction accuracy of multivariate nonlinear systems, even financial markets, and can help us to examine whether the structure of a complex system dynamically changes or not.

Keywords: nonlinear prediction, metaheuristic algorithms, coupled map lattice, financial markets

1. Introduction

There are many multivariate systems in the real world, such as financial markets, and their elements interact with each other and show complex behavior. To predict this behavior, we can make a prediction model based on the historical behavior observed from a complex system. However, although we can observe many elements as time-series data, these elements do not always compose the same system, and we do not know which observed elements actually interact with each other. Namely, it is difficult to identify the essential elements for prediction even if we can observe many elements. If we use irrelevant elements for making a prediction model, the model becomes complex and its generalization ability declines because of overfitting for learning data [1,2], and then prediction accuracy degrades.

For the above-described motivation, we could estimate interactions from the viewpoint of the similarity of each element’s behavior with several statistical measures: for example, the correlation coefficient, the partial correlation coefficient, the transfer entropy [3], Granger causality [4,5], and random matrix approach [6]. In our previous studies [7,8], we reported that the prediction model based on the estimated interaction can improve prediction accuracy. However, those studies also revealed that these measures are often ineffective in estimating interactions because of large nonlinearity or large synchronization of systems. In the present study, because estimating interactions might be difficult and is an indirect method of prediction, we directly select essential elements from among the observed elements so as to improve the prediction accuracy of learning data. Thus, estimating interactions is not a purpose of the present study.

First, we make prediction models using each combinatorial pair of elements, and then we estimate each prediction accuracy. Next, we find the optimal combinatorial pair that makes prediction accuracy the best. However, because this numerical cost grows exponentially with the number of elements, this identification is a combinatorial optimization problem. Although it is almost impossible to find the exact optimum solution, there are several metaheuristic searches [9] for
finding very good near-optimum solutions in a reasonable computational time. In the present study, we apply the genetic algorithm (GA) as an example of metaheuristic techniques, because the GA is one of the most popular techniques, and it is the first step in confirming whether the purpose of our study can be achieved using a standard metaheuristic algorithm. Moreover, each prediction model is coded as a binary series introduced in Section 2. This is also the reason why we use the GA, which enables easy handling of binary series.

As another method of reducing the numerical cost of metaheuristic algorithms, we can perform predictions merely by the single time-series data of an element that we want to predict, reconstructing an attractor of a multidimensional system in accordance with Takens’s embedding theorem [10, 11], as described in Section 2.2. However, we must optimize some parameters: the embedding dimension d and delay time of embedding τ, as introduced in Section 2.2.1. In previous studies, d was optimized by the false-nearest-neighbors (FNN) method [12–14] and τ by the autocorrelation function [15], but the main purpose of these methods is to reconstruct an attractor properly, not to predict it. However, as long as our main purpose is prediction, it is more plausible to directly optimize d and τ, from the viewpoint of prediction accuracy. Namely, we find the optimal parameters that yield the best prediction accuracy by applying a metaheuristic algorithm, as introduced in Section 2.2.

Moreover, it is possible that the mechanism of real complex systems is dynamically changed by external effects. In the example of financial systems, international accidents, politicians’ interviews, and daily news may change the mechanism of systems. In this case, an optimized prediction model worsens gradually, and therefore, we must iterate the optimization dynamically to make the prediction model more adaptable to a new changed system, as introduced in Section 2.3. In Section 3, we describe some simulations to examine the efficiency of this dynamical optimization, where foreign exchange markets are predicted as examples of real complex systems and artificial complex systems using the coupled map lattice (CML) [16]. Moreover, on the basis of the improvement of prediction accuracy, we examine whether these systems dynamically change or not.

2. Nonlinear Prediction of Multivariate Complex Systems

2.1 Multivariate prediction model

2.1.1 Using all observed elements

We observe a multivariate complex system and obtain each set of time-series data of N elements. Here, we denote a time-series datum of element i as \( x_i(t) \) and the temporal behavior of the system as

\[
v(t) = \{x_1(t), x_2(t), \cdots, x_i(t), \cdots x_N(t)\}
\]

In our study, to predict the future behavior of \( v(t) \), a simple prediction method is more desirable because we must save on the numerical cost of the genetic algorithm. Although there are many nonlinear prediction methods [17–22], we use the local linear approximation method [17] proposed by Lorenz in Eq. (2). First, we find some near neighbors \( v(t_k), k = 1, \cdots, K \), from the previous behavior, that is, \( t - L \leq t_k < t \). Here, we denote the number of near neighbors as \( K \) and the length of learning data as \( L \). The number of neighbors \( K \) should be set so as to maintain the local property of \( v(t) \) and to surround \( v(t) \) with \( v(t_k) \). The index \( K \) often depends on the dimension of the space of \( v \) and therefore, we set \( K = N + 1 \). Next, we predict the 1-step future behavior of \( v(t) \) by averaging each future behavior of neighbors \( v(t_k + 1) \):

\[
\bar{v}(t + 1) = \frac{1}{K} \sum_{k=1}^{K} v(t_k + 1)
\]

If we target only element \( i \) for prediction, we can obtain its predicted value \( \bar{x}_i(t + 1) \) from the i-th element of \( \bar{v}(t + 1) \). However, this prediction model does not always work well because all observed elements do not always compose the same system like Eq. (1). Namely, there is no guarantee that these elements are related to the target element \( i \). In our study, we consider two types of prediction models to avoid this inconvenience.

2.1.2 Optimizing essential elements for prediction with the GA algorithm

The first prediction model uses only essential elements for prediction, selected by a metaheuristic algorithm. In this paper, we apply the GA algorithm, as mentioned in Section 1.

If the target element \( i \) interacts with elements \( j \) (\( j = 1, 2, \cdots, n_i \)) strongly, it would be better to modify \( v(t) \) of Eq. (1) into

\[
v_i(t) = \{x_i(t), x_{i_1}(t), x_{i_2}(t), \cdots, x_{i_{n_i}}(t)\}
\]

and perform 1-step prediction by Eq. (2). Here, we set \( K = n_i + 1 \). This prediction model can yield a better selection of neighbors \( v(t_k) \) and can improve the prediction accuracy more than by using Eq. (1), which causes overfitting to the structure unrelated to element \( i \) and decreases the generalization ability [1,2] for new predictions after learning data.

In our study, to find a good multivariate prediction model such as Eq. (3), we apply the GA and find the optimal combinatorial pair that makes the prediction accuracy the best. Here, the number of
combinatorial pairs $C$ for making $v_i(t)$ grows exponentially to the number of observed elements $N$, that is, $C = 2^N$. Namely, this is a type of combinatorial optimization problem of searching for the optimum prediction model from among these combinatorial pairs. Therefore, we apply the GA to moderate this problem as follows.

**Step 1** Initialization of genotypes of the GA:
We initialize each genotype of the GA as a random binary series such as $g_i = \{11001 \cdots \}$. Here, $g_i(j) = 0$ means that element $j$ is not used to predict the target element $i$ as $v(t)$ of Eq. (3), and $g_i(j) = 1$ means that element $j$ is used. In our study, we prepared 30 genotypes.

**Step 2** Evaluation of the goodness of each genotype:
To evaluate the goodness of each genotype $g_i$, we estimate its prediction accuracy by predicting the previous behavior of element $i$, as introduced in Section 2.1.1. Here, to avoid overfitting by overfitting for learning data, we apply the cross-validation (CV) method [23–25], which is one of the resampling schemes and is often used for time-series prediction [26, 27]:

**Step 2-1** Preparing learning data and testing data:
We divide the previous behavior of element $i$ into two parts: $x_i(T_1), t-2L < T_1 \leq t-L$, as learning data and $x_i(T_2), t-L < T_2 \leq t$, as testing data, where $t$ means the present time.

**Step 2-2** Making trajectories based on genotype:
We create $v_i(T_1)$ and $v_i(T_2)$ on the basis of the genotype $g_i$.

**Step 2-3** Performing nonlinear prediction:
We search near neighbors from $v(T_1)$, and predict testing data $v(T_2)$ by the local linear approximation method introduced in Section 2.1.1. Then, we obtain predicted values $\tilde{x}_i(T_2)$.

**Step 2-4** Estimating each prediction accuracy:
We estimate prediction accuracy $\lambda_{1,i}$ using the difference between the true testing data $x_i(T_2)$ and the predicted values $\tilde{x}_i(T_2)$. The details are mentioned in Section 3.1. Similarly, by replacing the learning part and the testing part, that is, predicting learning data with testing data, we can obtain the next prediction accuracy $\lambda_{2,i}$.

**Step 2-5** Averaging each prediction accuracy:
We calculate the mean of $\lambda_{1,i}$ and $\lambda_{2,i}$ as $\lambda_i$, which gives a more reliable fitting accuracy without overfitting to the learning data. Also, we considered this fitting accuracy as the goodness $\lambda_i$ of the $i$-th genotype $g_i$.

**Step 3** Applying the GA algorithm:
Through each generation, each parent genotype $g_i$ can breed new children genotypes and can transfer its own properties to the next generation in accordance with its goodness $\lambda_i$.

**Step 3-1** Selecting parent genotypes:
Because better genotypes having higher goodness more easily survive and breed children, we preferentially select genotypes at random with replacement so that better genotypes are selected more frequently in accordance with $\lambda_i$. The number of selected genotypes is the same as that of initial genotypes.

**Step 3-2** Elitist selection:
In particular, the selected genotypes having the highest goodness are treated as elite genotypes and are cloned as children genotypes without any modification. In our study, we treated 10% of the selected genotypes as elite genotypes, and distributed the other genotypes to crossover and mutation randomly at the ratio of 9:1.

**Step 3-3** Crossover:
Crossover is used to breed two genotypes. If two genotypes are $g_1 = \{00000\}$ and $g_2 = \{11111\}$, new genotypes are, for example, $g_1' = \{10101\}$ and $g_2' = \{01010\}$, which become parent genotypes of the next generation. This crossover is called multipoint crossover, where crossover points are decided randomly.

**Step 3-4** Mutation:
Mutation is used to avoid genotypes becoming too closely related to each other. Mutation points are selected randomly, and each gene of selected mutation points is changed. If the value is 1, it is replaced by 0.

**Step 4** Iterating generational change:
Through step 3, we can create genotypes of a new generation, and return these genotypes to step 2. By iterating steps 2 and 3, we can evolve each genotype, and can obtain a good near-optimum solution in a reasonable computational time. In our study, the number of generations is 500.

**Step 5** Predicting unknown future behavior using the optimized prediction model:
After sufficient evolution of the GA, we can find the best genotype $g^*$ with the largest goodness $\lambda_i$, and can optimize the prediction model of Eq. (3) with $g^*$, which is denoted as $v^*_i$. Then, we predict the future behavior of $x_i$ using $v^*_i$ in Eq. (2).
2.2 Single-variate prediction model

The second prediction model uses only a target element for prediction, and reconstructs an attractor of a complex system on the basis of Takens’s embedding theorem [10,11]. Recently, although more sophisticated embedding methods [28,29] have been proposed, Takens’s embedding theorem is very simple and is useful for saving on the numerical cost of the GA. Moreover, we need not consider any interaction among elements because we can substitute delayed time-series data of the single element for multivariate elements interacting with it. A reconstructed attractor is as follows:

\[ \mathbf{v}_i(t) = \{x_i(t), x_i(t - \tau), \cdots, x_i(t - (d - 1)\tau)\} \] (4)

where \( d \) means the embedding dimension and \( \tau \) means the delay time. Then, we set \( K = d + 1 \), and we predict \( \mathbf{v}_i(t + 1) \) using Eq. (2) to average the future behavior of neighbors \( \mathbf{v}_i(t_k + 1) \). Here, it is most important that \( \mathbf{v}_i \) should be reconstructed using appropriate \( d \) and \( \tau \) in Eq. (4) to realize the best prediction.

2.2.1 Conventional methods of optimizing \( \tau \) and \( d \)

Each component of the embedded attractor \( \mathbf{v}_i(t) \) corresponds to each dimensional coordinate of the state space. To orthogonalize each coordinate, we must minimize the correlation among components of \( \mathbf{v}_i(t) \). Therefore, it is popular to set the delay time \( \tau \) as the time when the autocorrelation function of \( x_i(t) \) becomes 0 first, or becomes the minimum value if the autocorrelation function does not become 0 [15].

Moreover, to optimize the embedding dimension \( d \), the false-near-neighbors (FNN) method [12-14] is useful. The method is used to determine whether each nearest neighbor \( \mathbf{v}(t_1) \) is a false neighbor or not, by increasing the embedding dimension \( d \). If the nearest neighbor is separated from \( \mathbf{v}(t) \) upon increasing \( d \), the neighbor is considered false neighbor. Therefore, we can optimize \( d \) so as to avoid such false neighbors. If \( d \) is more than the optimum value, there is no false neighbor.

In this method, we denote the distance between \( \mathbf{v}(t) \) and \( \mathbf{v}(t_1) \) as \( D_d(t) \) in \( d \)-dimensional state space, and denote it as \( D_{d+1}(t) \) in \( (d + 1) \)-dimensional state space. Their relative distance \( D_L(t) \) is defined by

\[ D_L(t) = \sqrt{\frac{D_{d+1}^2(t) - D_d^2(t)}{D_d^2(t)}} \] (5)

If \( D_L(t) \) is more than 15, \( \mathbf{v}(t_1) \) is considered to be the false neighbor of \( \mathbf{v}(t) \) [12].

To optimize \( d \), we calculate each \( D_L(t') \) (\( t - L \leq t' \leq t \)), where the nearest neighbor \( \mathbf{v}(t') \) is searched from \( t - L \leq t' < t \). Then, we set \( d \) so that the rate of false neighbors is less than 5% of the total \( L \).

2.2.2 Our optimization of \( \tau \) and \( d \) with the GA algorithm

Although the FNN method and the autocorrelation function are useful for deciding \( d \) and \( \tau \), we can apply the GA as in Section 2.1.2 to optimize them. This optimization is also considered as a type of combinatorial optimization problem. If each parameter is described by \( M \) bits, the number of combinatorial pairs \( C \) for making \( \mathbf{v}_i(t) \) is \( 2^{2M} \). If we set \( 2M < N \), we can reduce the numerical cost of the GA because \( C \) in Section 2.1.2 is \( 2^N \). The procedure for optimizing \( \tau \) and \( d \) is as follows.

**Step 1** Initialization of genotypes of the GA:

Similarly as in Section 2.1.2, we initialize each genotype of the GA as random binary series. However, each genotype represents a set of \( d \) and \( \tau \). If a genotype is described by 10 bits such as \( g = \{11001 \cdots 0\} \), the first half and the last half correspond to \( d \) and \( \tau \), respectively. That is, \( M = 5 \).

**Step 2** Evaluation of the goodness of each genotype:

This step is almost the same as that in Section 2.1.2. However, in Step 2-2, we apply Eq. (4) to create \( \mathbf{v}_i(T_1) \) and \( \mathbf{v}_i(T_2) \) on the basis of genotype \( g \).

**Step 3** Applying the GA algorithm:

This step is the same as that in Section 2.1.2.

**Step 4** Iterating generational change:

This step is the same as that in Section 2.1.2.

**Step 5** Predicting unknown future behavior with the optimized prediction model:

This step is almost the same as that in Section 2.1.2, but we optimize the attractor \( \mathbf{v}_i^* \) of Eq. (4) with \( g^* \). Then, we predict the future behavior of \( x_i \) using \( \mathbf{v}_i^* \) in Eq. (2).

2.3 Dynamical optimization

In the above subsections, we introduced four types of prediction methods. Let us denote each method as follows.

**Method 1:**

Use all observed elements for Eq. (1), as in Section 2.1.1 without any parameters or any optimization.

**Method 2:**

Select elements with the GA for Eq. (3), as in Section 2.1.2

**Method 3:**

Optimize embedding parameters of Eq. (4) by conventional methods, the FNN method and the autocorrelation function, as in Section 2.2.1,
3. Numerical Simulations

In this section, to examine the efficiency of Methods 1 ~ 4 and their dynamical optimizations, we perform several numerical simulations. However, Method 1 does not apply dynamical optimization because it always uses all of the observed elements.

3.1 Multivariate complex systems for simulations

With the first simulation, we predict an artificial model of complex systems. The model is based on the coupled map lattice (CML) [16] and is denoted by

\[ x_i(t + 1) = F \left( (1 - \epsilon)x_i(t) + \frac{\epsilon}{n_i} \sum_{j \in G_i(p)} x_j(t) \right) \]  

(6)

where \( i \) is the index of elements (\( i = 1, 2, \cdots, N \)), \( n_i \) is the number of elements interacting with the \( i \)-th element, \( G_i(p) \) is the set of these elements, and \( \epsilon \) is the strength of their interaction with the \( i \)-th element.

Although the regular CML [16] has a ring lattice where each element interacts with only its two neighbors, this CML of Eq. (6) is modified by the WS model [30] so that the topology \( G_i(p) \) of interaction can be changed according to \( p \) [31]. In making the WS model, first, we prepare a ring lattice where each node is connected with the nearest \( k \) neighbors; that is, we prepare a regular network [30], and the total number of edges is \( \frac{kN}{2} \). In our study, we set \( N = 100 \) and \( k = 4 \). Next, we rewire each edge according to the rewiring probability \( p \), cutting one side of an edge and randomly connecting it to some node. Additionally, once rewired, edges are fixed. If we set \( p = 0 \), the topology \( G_i(p) \) is the regular network. Then, we can realize the small-world network [30] by setting \( 0 < p < 1 \) and the random network [30] by setting \( p = 1 \). In our study, we set \( p = 1 \) because the topology of the random network is the most complex. Moreover, as an example of dynamics \( F \), we use the logistic map, \( F(x_i) = 1 - \alpha x_i^2 \), which is widely analyzed as a type of chaotic map. If \( \alpha = 2 \), each element moves chaotically owing to the dynamics of the logistic map. Therefore, we set \( \alpha = 2 \) and \( \epsilon = 0.3 \) so as to derive complex behavior.

Figure 4(a) shows the whole behavior of this modified CML, and we can confirm that this behavior is highly complex. Then, the maximum Lyapunov exponent [32,33] of this system is estimated to be 2.25. Namely, we can check that this behavior was derived from chaotic nonlinear dynamics because the maximum Lyapunov exponent is positive. However, the mechanism of this system does not change temporally, unlike real systems, because the topology \( G_i(p) \) is fixed in Eq. (6).

At the second simulation, we tried predicting real foreign-exchange markets (Forex) in 1996 [34]. As in-
Fig. 2 Flowchart of Method 2 or 4

Fig. 3 Flowchart of Method 3
Fig. 4 (a) Movement of modified CML of Eq. (6), (b) Price movement of 25 foreign-exchange (Forex) markets in 1996: Details are mentioned in Section 3.1. For better visualization, we show the modified price movement $\tilde{x}_i(t)$ by the function $	ilde{x}_i(t) = \log(1 + x_i(t))$ if $x_i(t) \geq 0$, otherwise $\tilde{x}_i(t) = -\log(1 - x_i(t))$, where $x_i(t)$ is an original price movement of Forex $i$. (c) Macrograph of (b) from $t = 250$ to $t = 650$.

3.2 Prediction results

Figures 5 and 6 show the prediction accuracy $\lambda_i[\%]$ of the modified CML of Eq. (6) with $L = 1,000$ in each figure (a), and show that of the 25 Forex markets with $L = 100$ in each figure (b). Here, $L$ was adjusted so as to predict each system more accurately.

Then, to compare each set of $\lambda_i$ obtained by each method, we performed the paired t-test; Figure 5 shows the significant improvement of Method 2 against Method 1, and Fig. 6 shows that of Method 4 against Method 3. Here, if the difference between these two methods is confirmed by a significance level of $\alpha = 0.05$ or $\alpha = 0.01$, we marked it with a triangle or a square on the plot, respectively. From these results, we can conclude that using metaheuristic optimization is very useful for improving not only the multivariate prediction model of Eq. (3) but also the single-variate prediction model of Eq. (4), especially when predicting the 25 Forex markets composing a real complex system.

Moreover, only when we predict the 25 Forex markets, we can confirm that prediction accuracy becomes better by using larger $\theta$, so that the optimization is performed more frequently. From the result, we can infer that the 25 Forex markets do not compose a stationary system and change their individual mechanisms dynamically. To predict such systems, the dynamical optimization would be effective. On the other hand, it would be ineffective for the modified CML whose mathematical structure is constant as defined by Eq. (6).

Next, to examine the advantage of the dynamical optimization against the statical optimization more
Table 1 shows the two sets of prediction accuracies $\{\lambda_i\}$ obtained by the statical prediction when $\theta = \theta_1$ and by the dynamical prediction when $\theta = \theta_2$, as shown in Figs. 5 and 6. Each paired t-test also shows the advantage of the dynamical optimization for only the prediction of the 25 Forex markets by meta-heuristic optimization, that is, Method 2 and Method 4. These results also support the fact that this real system changes dynamically, and the dynamical optimization can detect such a dynamical change of real systems.

Moreover, although the prediction accuracy of the modified CML is better with larger $L$, that of the Forex markets is better with smaller $L$. If a system changes dynamically, the large amount of learning data often includes some different kinds of structures and worsens the prediction accuracy. Namely, this viewpoint also supports the dynamical change of real Forex markets.

Fig. 5 Prediction accuracy $\lambda_i[\%]$ of (a) modified CML of Eq. (6) with $L = 1,000$, and (b) 25 Forex markets with $L = 100$: Each solid line shows the mean value of prediction accuracies $\langle \lambda_i \rangle = \frac{1}{N} \sum_{i=1}^{N} \lambda_i$, and each error bar shows their standard deviation $\sigma_{\lambda_i}$. Thin line shows the results of Method 1 using all of the observed elements for Eq. (1) in Section 2.1.1, and thick line shows those of Method 2 with optimized elements by the GA for Eq. (3) in Section 2.1.2. If the difference between these methods is confirmed by the paired t-test at the significance level of $\alpha = 0.05$ or $\alpha = 0.01$, we marked a triangle or a square, respectively, on the plot. The $\theta_1$ of the horizontal axis means the statical optimization, that is, the optimization is performed only once at the first prediction, and $\theta_2$ means the dynamical optimization, that is, the optimization is iterated every prediction. Because Method 1 uses all of the observed elements, it does not depend on $\theta$.  

Fig. 6 Same as Fig. 5, but thin line shows the results of Method 3 with adjusted embedding parameters of Eq. (4) by conventional methods described in Section 2.2.1, and thick line shows those of Method 4 with optimized embedding parameters of Eq. (4) by the GA in Section 2.2.2.
Table 1  Difference between the statical prediction when \( \theta = \theta_1 \) and the dynamical prediction when \( \theta = \theta_2 \) in Figs. 5 and 6: Each \( (\lambda_i) \) is the mean value of prediction accuracies \( \lambda_i(\%, i = 1, 2, \cdots, N. \) Because Method 1 cannot be applied for dynamical optimization, the results of Method 1 are omitted. If prediction accuracy \( \lambda_i \) of each element is improved by dynamical optimization, we counted these elements in the improving ratio. On the other hand, if the dynamical prediction degrades \( \lambda_i \) of each element, we counted these elements in the degrading ratio. Then, by the paired \( t \)-test of significance, if we can confirm the difference between the dynamical and the statical optimizations, the significance level \( \alpha \) is shown; if we cannot confirm the difference, "---" is marked.

<table>
<thead>
<tr>
<th>Optimization</th>
<th>Method 2</th>
<th>Method 3</th>
<th>Method 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Predicted data</td>
<td>CML</td>
<td>Forex</td>
<td>CML</td>
</tr>
<tr>
<td>( (\lambda_i) ) when ( \theta = \theta_1 )</td>
<td>90%</td>
<td>56%</td>
<td>97%</td>
</tr>
<tr>
<td>( (\lambda_i) ) when ( \theta = \theta_2 )</td>
<td>92%</td>
<td>60%</td>
<td>96%</td>
</tr>
<tr>
<td>Improving ratio</td>
<td>63%</td>
<td>96%</td>
<td>33%</td>
</tr>
<tr>
<td>Degrading ratio</td>
<td>46%</td>
<td>4%</td>
<td>67%</td>
</tr>
<tr>
<td>Significance test</td>
<td>---</td>
<td>( \alpha = 0.01 )</td>
<td>---</td>
</tr>
</tbody>
</table>

4. Conclusion

In our study, we compared four types of prediction methods: all of the observed elements for Eq. (1) are used in the first method; the elements optimized by the genetic algorithm (GA) for Eq. (3) are used in the second method; the embedding parameters of Eq. (4) adjusted by conventional methods are used in the third method; and the embedding parameters optimized by the GA used in the fourth method. Moreover, we examined the efficacy of their dynamical optimizations.

The main concept of our study is to apply meta-heuristic techniques to improve basic prediction methods: a multivariate prediction model or a single-variante prediction model to optimize each prediction accuracy. Moreover, to save on the numerical cost of metaheuristic optimization, we used a simple local linear prediction, which is one of the nonlinear predictions, for embedded attractors. In spite of the simple prediction, we can realize improved prediction accuracy of complex systems efficiently, even if their mechanism often changes dynamically. Moreover, we demonstrated the identification of whether complex systems dynamically change or not, using the statistical significance test based on the difference between the prediction accuracies in the cases of dynamical optimization and statical optimization.

In a future study, we can apply different metaheuristic algorithms, such as the simulated annealing algorithm and the particle swarm optimization, and we believe it is important to clarify the best algorithm for optimizing prediction models from the viewpoints of prediction accuracy and numerical cost.

This research was partially supported by the Grant-in-Aid for Young Scientists (B) Grant No. 22700227 from the Ministry of Education, Culture, Sports, Science and Technology of Japan.

References


[34] We purchased the set of historical data from Olsen Corp. (http://www.olsendata.com/).


Tomoya Suzuki received his B.S., M.S., and Ph.D. degrees in Physics from Tokyo University of Science, Tokyo, Japan, in 2000, 2002, and 2005, respectively. He joined Tokyo Denki University as an Assistant in 2005, and was a Lecturer in Doshisha University from 2006 to 2009. Since 2009, he has been an Associate Professor in Ibaraki University, Japan. His previous interests include complex systems, financial systems, time series analysis, prediction, and data mining. He is a member of IEICE, IPSJ, the Physical Society of Japan (JPS), and the Nippon Technical Analysts Association (NTAA).

(Received May 12, 2012; revised August 1, 2012)