ELSEVIER

Fuzzy Sets and Systems 87 (1997) 265–276

FUZZY sets and systems

Real-valued genetic algorithms for fuzzy grey prediction system

Yo-Ping Huang*, Chih-Hsin Huang

Department of Computer Science and Engineering, Tatung Institute of Technology, Taipei, Taiwan 104, ROC

Received June 1995; revised December 1995

Abstract

A genetic-based fuzzy grey prediction model is proposed in this paper. Instead of working on the conventional bit by bit operation, both the crossover and mutation operators are real-valued handled by the presented algorithms. To prevent the system from turning into a premature problem, we select the elitists from two groups of possible solutions to reproduce the new populations. To verify the effectiveness of the proposed genetic algorithms, two simple functions are first tested. The results show that our method outperforms the conventional one no matter whether from the viewpoint of the number of iterations required to find the optimum solutions or from the final solutions obtained. The real-valued genetic algorithms are then exploited to optimize the fuzzy controller which is designed to perform the compensation job. Two different types of fuzzy inference rules are considered to compensate for the predicted errors from the grey model. The difficulty encountered in applying the genetic algorithms to adjusting the fuzzy parameters is also discussed. Based on the simulation results from the problems of the weather forecast, we found that the proposed methodology is very effective in determining the quantity of compensation for the predicted outputs from the traditional grey approach. © 1997 Elsevier Science B.V.

Keywords: Genetic algorithms; Fuzzy grey system; Fuzzy inference models

1. Introduction

Recently, genetic algorithms have been widely applied to different optimization problems [3–5]. Unlike the conventional algorithms which optimize the problems from a single direction, the genetic algorithms search for the optimal solutions from multiple directions. This allows the genetic-based method to have a better chance of finding the optimal or near-optimal solutions. Besides, due to the simplicity in programming, the genetic algorithms can be considered to replace the gradient descent method [6, 9] to automatically optimize both the parameters and the structure in the fuzzy system.

In order to search for the optimal solution for a given problem, a fitness function is defined to evaluate the performance for each chromosome. The well-known roulette wheel selection criterion [4] is adopted here to decide whether a chromosome can survive or not in the next generation. The survival chromosomes are then put into a mating pool for the crossover and mutation operations. Once a pair of strings have been selected for crossover, a randomly selected site is assigned into the to-be-crossed strings. One substring then exchanges part of its string (from the crossover site to the end of the string) with the other's. The newly-crossed strings join the rest of the chromosomes to form a new population. The mutation operation follows the crossover to offer a chance for each bit to flip. The evolution continues until some preset criteria are met.

* Corresponding author.

PII S0165-0114(96)00011-5

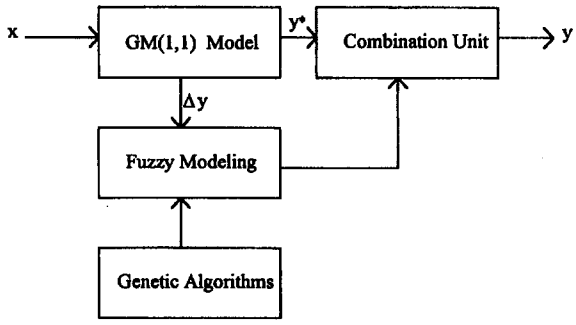

Fig. 1. The block diagram of the genetic-based fuzzy grey model. Note that input x to the grey model corresponds to the just past outputs from the unknown plant. y^* and Δy refer to the predicted output and predicted error from grey model, respectively.

To use the genetic algorithms for the optimization problems, all of the parameters or variables involved in the problem are first binary encoded to form a chromosome, which corresponds to a possible solution to the problem in the searching space. Note that the necessity for binary encoding the required parameters is a controversial issue [3]. In this paper, we will handle the basic operators in the form of real values. Two novel algorithms are presented in Section 3.

We then apply the genetic algorithms to the fuzzy grey prediction problem. Fuzzy logic has been well-recognized for its suitability in the control areas [10, 11]. Grey theory was first proposed by Deng in 1982 [1, 2], 17 years after Zadeh's fuzzy set theory [12] was introduced. Grey theory is based on the grey hazy sets [1] which differ from the fuzzy sets. We first integrated these two modeling methods into the prediction problem [7, 8]. In [7, 8], the data used in forming the grey model become the antecedent part in the fuzzy rule and the predicted error from the grey model belongs to the consequent part. The purpose is to apply the genetic algorithms to automatic adjustment of the membership functions to satisfy the given data pairs. The well-tuned model is then used for the current input pattern to obtain the quantity of compensation for the predicted output from the grey model.

In summary, this paper focuses on the following three themes: (1) establishing a grey model to predict the next output from an unknown plant; (2) utilizing the given data to design a fuzzy controller to remedy the prediction error from the grey model; and (3) applying the proposed genetic algorithms to the optimization of the fuzzy rule bases. The structure of the presented fuzzy grey system is plotted in Fig. 1. The GM(1, 1) block is dedicated to predicting the system's next output based on the recently past outputs. The fuzzy model in Fig. 1 is adopted to modify the predicted result from the grey model such that the system can reach the desired outcome. The block of genetic algorithms is applied to the optimization of the fuzzy rule base.

The remainder of this paper is organized as follows. To provide the readers who might not be well acquainted with the grey theory, we briefly introduce the basic concepts of grey theory in Section 2. The real-valued genetic algorithms are proposed in Section 3. To illustrate the applicability of the presented algorithms, two simple functions are simulated in Section 4. In Section 5, we discuss how the fuzzy logic and grey system can be integrated into a hybrid model to predict the temperatures. The way to apply genetic algorithms to the adjustment of the membership functions is also stated. Simulation results from predicting the monthly average temperatures for two different cities are given in Section 6. In Section 6, we also discuss why the optimal parameters for fuzzy models are hard to find with the genetic algorithms. Conclusion is made in the final section.

2. The basic concepts of grey system

The predicted output from an unknown plant can always provide us with some important information to better control the system. However, the accuracy of the predicted value depends heavily on the constructed approximated model. A grey system is a system with characteristics between white and black ones. This implies that the grey system can deal with a system which consists of partially known and partially unknown information. Instead of analyzing the characteristics of the unknown system mathematically, the grey system exploits the accumulated generating technique to approach the plant's behavior. The raw data output from the plant may not possess any regularity. However, the original data may become more regular after a repeatedly accumulated generating operation [1]. Therefore, we can use the differential equation to approximate such a regular-

ity and hopefully to predict the next output from the plant.

Since the grey system stresses that it can deal with a system with insufficient information contained, a grey model can be established as long as there are four past data available. The advantages of the grey model are: (1) the next output from an unknown plant can be predicted based on a few past output data, and (2) it can use a first order differential equation to characterize the unknown system behavior [1, 7]. That is, we can use a few discrete data to form a first order differential equation to characterize such an unknown system. Whenever a new datum is received, this new datum joins the original data sequence by removing the oldest datum in the original data sequence such that a new grey model can be formed immediately. Thus, the grey system can be updated quickly to satisfy the prediction problem in time-varying nonlinear system.

The most commonly used grey model is the GM(1, 1) model, i.e., a single variable first order grey model. We summarize the modeling procedures as follows [1]:

Step 1: Given the initial data $X(0) = [x(1), x(2), \ldots, x(n-1), x(n)]$, where $x(i)$ corresponds to the system output at time i, we try to predict the next $x(n+k)$, $k \geqslant 1$.

Step 2: From the initial $X(0)$ a new sequence $X(1)$ is generated by the accumulated generating operation (AGO), where $X(1) = [x^1(1), x^1(2), \ldots, x^1(n)]$ and is derived as follows:

$$x^1(k) = \sum_{m=1}^{k} x(m). \tag{1}$$

Step 3: From $X(1)$ we can form the following first order differential equation:

$$\frac{\mathrm{d}x^1}{\mathrm{d}t} + ax^1 = u. \tag{2}$$

Step 4: From Step (3) we have

$$\hat{x}^1(k+1) = \left(x(1) - \frac{u}{a}\right) \mathrm{e}^{-ak} + \frac{u}{a}, \tag{3}$$

$$\hat{x}(k+1) = \hat{x}^1(k+1) - \hat{x}^1(k), \tag{4}$$

where

$$\hat{a} = \begin{bmatrix} a \\ u \end{bmatrix} = (B^{\mathrm{T}}B)^{-1}B^{\mathrm{T}}y_N, \tag{5}$$

$$B = \begin{bmatrix} -0.5(x^1(1) + x^1(2)) & 1 \\ -0.5(x^1(2) + x^1(3)) & 1 \\ \cdots & \cdots \\ -0.5(x^1(n-1) + x^1(n)) & 1 \end{bmatrix},$$

$y_N = [x(2), x(3), \ldots, x(n)]^{\mathrm{T}}$, and $\hat{x}(k+1)$ is the predicted value of $x(k+1)$ at time $k+1$.

The grey relational space (GRS) is originally believed to have captured the relationship between the main factor and the other reference factors in a given system [1]. We apply this technique here to relate the current modeling pattern to the previous patterns which show some degrees of similarity to the current one such that the quantity of adjustment to the predicted outcome can be determined. The grey relational model can be summarized as follows:

Step 1: Let the reference sequence be $x_0 = (x_0(1), x_0(2), \ldots, x_0(n))$.

Step 2: Denote the m sequences to be compared with by $x_i = (x_i(1), x_i(2), \ldots, x_i(n))$, $i = 1, 2, \ldots, m$.

Step 3:

$$\gamma(x_0(k), x_i(k)) = \frac{\min_j \min_k |x_0(k) - x_j(k)|}{|x_0(k) - x_i(k)| + \zeta \max_j \max_k |x_0(k) - x_j(k)|} + \frac{\zeta \max_j \max_k |x_0(k) - x_j(k)|}{|x_0(k) - x_i(k)| + \zeta \max_j \max_k |x_0(k) - x_j(k)|},$$

where $\zeta \in (0, 1]$ is the distinguishing coefficient. $j = 1, 2, \ldots, m$, $k = 1, 2, \ldots, n$. $\gamma(x_0(k), x_i(k))$ is called the grey relational coefficient at point k. From this grey relational coefficient we can obtain the grey relational grade.

Step 4: The grey relational grade is derived as follows:

$$\gamma(x_0, x_i) = \frac{1}{n} \sum_{k=1}^{n} \gamma(x_0(k), x_i(k)).$$

Here, $\gamma(x_0, x_i)$ represents to what degree of influence the sequence x_i can exert on the reference sequence x_0. In other words, the reference sequence can grasp some useful information about the variation of data points from other similar sequences. From the analysis of the grey relational grade we can understand which factors will crucially affect the reference factor. In the

sequel, we picked some most related patterns with the current one from the past to train the fuzzy controller. The well-trained fuzzy model was then used to decide the quantity of adjustment for the grey output.

3. Real-valued genetic algorithms

As we know, the genetic algorithm is based on the Darwinian principles of biological evolution. It works with the coding of the parameter set, not the parameters themselves. Besides, it optimizes the problem through a population of solutions, not just the single-point search [4]. When a set of possible solutions is generated under certain constraints, each candidate is usually binary coded into a vector chromosome before the three major operators can proceed.

The initial population of candidate solutions is usually randomly generated. However, a better selection of the initial population implies a quicker convergence to the optimum or near-optimum solution. The problem is how to initiate a good start. In this paper, we use the elite selection criterion to initiate the population. This approach, however, is quite different from the elitist strategy proposed by Grefenstette [5]. In Grefenstette's algorithm, the most fitted chromosome always survives intact into the next generation. Our approach starts with two randomly generated groups of possible solutions. This is done in the hope of widening the searching space while keeping the final population in a compact size. Each group is allowed to self evolve for a few generations before half of the best solutions from each group are selected to be combined into one brand new population set. These new populations become the initial possible solutions for the optimization problem. The block diagram of the two-group genetic algorithms is plotted in Fig. 2.

The problem existing in the binary coding lies in the fact that a long string always occupies the computer memory even though only a few bits are actually involved in the crossover and mutation operations. This is particularly the case when a lot of parameters are needed to be adjusted in the same problem and a higher precision is required for the final result. To overcome the inefficient occupation of the computer memory, the underlying real-valued crossover and mutation algorithms are proposed.

3.1. Fixed-point crossover algorithm

Denote the string length of the chromosome by LENGTH and the bit length of the coded parameter by BIN_BIT. Note that LENGTH corresponds to the number of parameters needed to be optimized in the problem. The procedures to perform the crossover operation are as follows:

Step 1: According to the preset probability of crossover (P_c), select the number of chromosomes to do the crossover.

Step 2: Randomly select a parameter to be crossovered by *random*(LENGTH) command.

Step 3: Select a position from the binary bits by *random*(BIN_BIT-1) command.

Step 4: Set index = *random*(BIN_BIT-1)

```
D1 = CODE1 % pow(2, index+1);
D2 = CODE2 % pow(2, index+1);
if (D1! = D2)
   CODE1 = CODE1−D1+D2;
   CODE2 = CODE2−D2+D1;
else
   CODE1 = CODE1;
   CODE2 = CODE2;
pair_of_crossover_chromosome
 = pair_of_crossover_chromosome-1;
```

Step 5: If pair_of_crossover_chromosome ! = 0, goto Step 2; else END.

Note that in the above algorithm, *random*(J) is a turbo C command, which will randomly generate a number between 0 and $J-1$. CODE1 and CODE2 are the pair of parameters under crossover operation. The symbol % is the modulus operator and pow(2, i) means 2^i. To better illustrate the algorithm mentioned above, the following example is given.

Example. Assume that three parameters, denoted by $(L, C, R)_{10}$ which corresponds to the three parameters characterizing a triangular membership function, are considered. Note that the subscript 10 means that the parameters are represented in real values. Since three parameters are considered, this means that LENGTH = 3. Suppose that each parameter requires 5 bits, i.e., BIN_BIT = 5, to encode it. As a result, in the binary coding scheme the string length of the

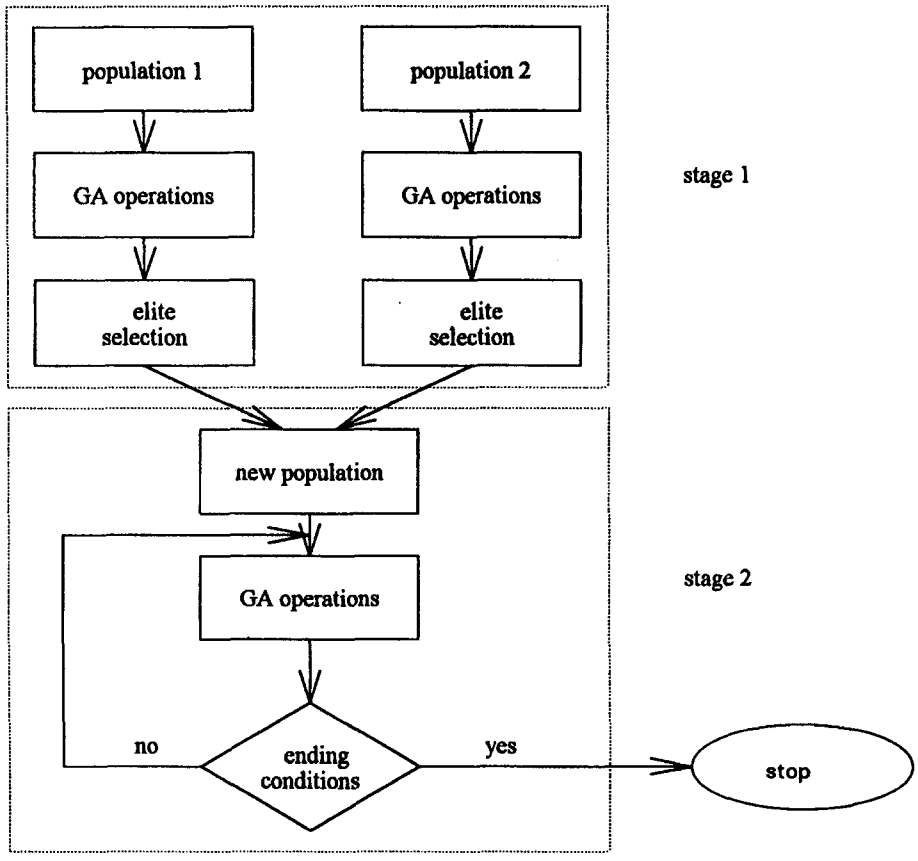

Fig. 2. The block diagram of the two-group genetic algorithms.

chromosome equals 15. Assume that a pair of chromosomes (the parents), i.e., $(24, 14, 24)_{10}$ and $(16, 1, 7)_{10}$, will crossover at the middle parameters and index = 1, which correspond to the site indicated by "|" in the following traditional expression:

$$\text{parent\_1} = (11000011 \mid 1011000)_2$$

and

$$\text{parent\_2} = (10000000 \mid 0100111)_2.$$

Note that the subscript 2 in the above representation means that the chromosome is binary coded. Under the conventional approach, the offsprings arising from the crossover operation will, respectively, become:

$$\text{offspring\_1} = (11000011 \mid 0100111)_2$$

and

$$\text{offspring\_2} = (10000000 \mid 1011000)_2,$$

which correspond to $(24, 13, 7)_{10}$ and $(16, 2, 24)_{10}$, respectively. As can be seen from the crossover operation, the first five bits in each string remain unchanged after crossover. The last five bits simply exchange with the other's. Only the middle parameters will have their values changed. In this example, the binary substring 01110 from parent_1 equals 14, i.e., CODE1 = 14, in the decimal expression. Also, CODE2 is 1 from the binary substring 00001 in parent_2. Since index = 1, D1 and D2 will be equal to 2 and 1, respectively. As a result, new CODE1 = $14 - 2 + 1 = 13$ and new CODE2 = $1 - 1 + 2 = 2$, which are exactly the middle decimal values as appearing in offspring_1

and offspring_2, respectively. Thus, in this paper each chromosome is treated as a cascade of real numbers.

3.2. Mutation algorithm

When a gene in a string is selected to be mutated, we just flip the bit from 0 to 1 and from 1 to 0. This means that only the selected parameter in the chromosome will change its value and the rest will remain unchanged. Based on this observation, the following mutation algorithm is presented:

Step 1: According to the preset probability of mutation (P_m), select the number of chromosomes to do the mutation.

Step 2: Randomly select a parameter to be mutated by *random*(LENGTH) command.

Step 3: Select a position from the binary bits by *random*(BIN_BIT) command.

Step 4: Set index = *random*(BIN_BIT)
if CODE % pow(2, index + 1)
< pow(2, index)
then CODE = CODE + pow(2, index);
else CODE = CODE − pow(2, index);
no_of_mutation_chromosome
=no_of_mutation_chromosome-1;

Step 5: If no_of_mutation_chromosome ! = 0, goto Step 2; else END.

Example. This example shows how the proposed real-valued mutation operation works. Suppose a chromosome $(24, 14, 24)_{10} = (11000, 01110, 11000)_2$ is to be mutated at position 8, i.e., at address 7, from the right-hand side in the corresponding binary string. This means that the middle parameter will be mutated and index = 2. Since both the rightmost and leftmost parameters will remain unchanged after mutation, only the middle parameter has to be considered. The middle binary substring is 01110, which corresponds to 14 in the decimal expression. After undergoing the mutation operation, the middle substring will become 01010, which is ten on the base of 10. This means that the new chromosome will become $(24, 10, 24)_{10}$. Our algorithm conforms to this explicit result. This is due to the fact that index = 2, CODE = 14, and (CODE % pow(2, index+1)) = 6 is not less than pow(2, index) = 4. Thus, the new CODE = 14 − pow(2, index) = 10. This further confirms that all the parameters can be stored in the real-valued forms instead of the binary strings.

4. Verification of the proposed genetic algorithms

Before applying the proposed genetic algorithms to the fuzzy grey system, two simple functions are first simulated to verify the effectiveness of the novel algorithms.

Example 1. Find the maximum of the following function [3]:

$$f = 100(x_1^2 - x_2)^2 + (1 - x_1)^2,$$

$$-2.048 \leqslant x_i \leqslant 2.048.$$

In this example, we set (population size, P_c, P_m) = (30, 0.53, 0.1). Each parameter is encoded into 12 bits when an operation has to be performed. We randomly generate two groups of populations possessing the conditions just described and each group self evolves for five generations before the elite operation is activated. One of the simulation results is plotted in Fig. 3. The evolution process from the conventional approach is also included in the same figure for comparison. It is obvious that our algorithm outperforms the traditional one no matter whether from the viewpoint of converging speed or from the final found value.

Example 2. Find the maximum of the following function:

$$g(u(k)) = 6 \sin(\pi u(k)) + 3 \sin(3\pi u(k)) + \sin(5\pi u(k)),$$

where $u(k) = \sin(2\pi k/250)$, $275 \leqslant k \leqslant 375$ and k = integer.

The conditions are set as (population size, P_c, P_m) = (30, 0.53, 0.2) in this case. Fig. 4 shows the simulation results of several trials based on both our algorithms and the traditional ones. Again, our approach outperforms to a great extent the conventional one.

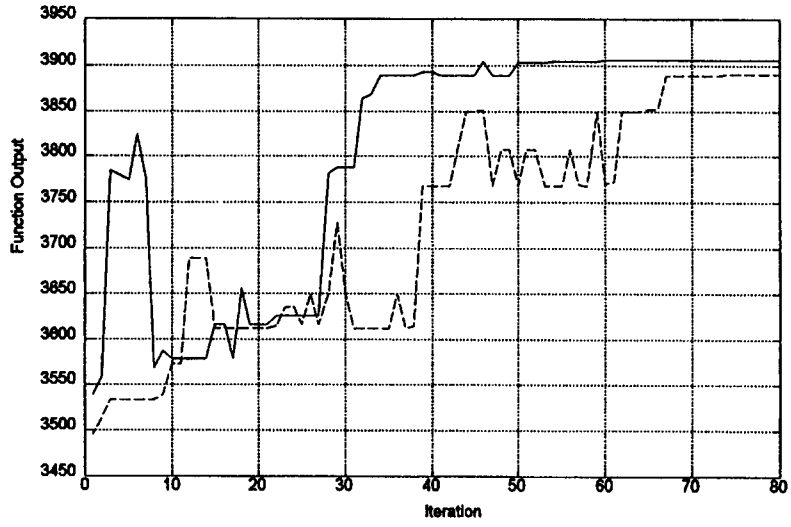

Fig. 3. Simulation results for Example 1. Note that solid and dashed curves represent outputs from our algorithms and the conventional ones, respectively.

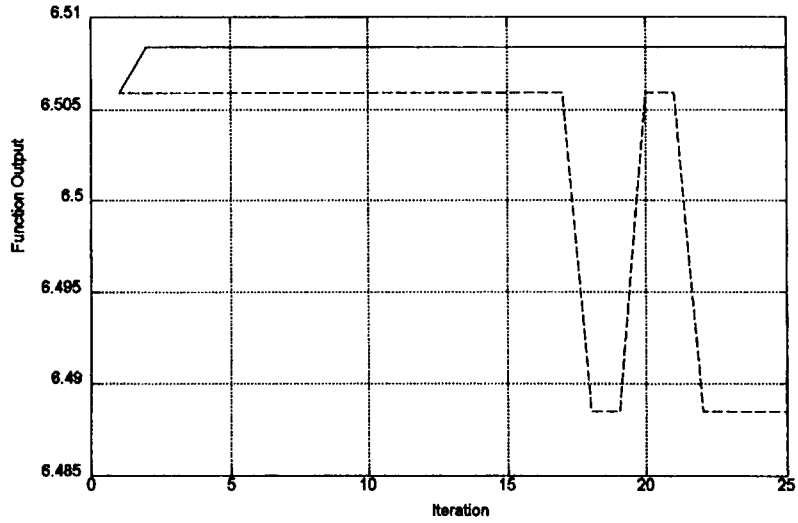

Fig. 4. Simulation results for Example 2. Note that solid and dashed curves represent outputs from our algorithms and the conventional ones, respectively.

5. The fuzzy grey system

The successful applications of the proposed real-valued genetic algorithms to the optimization problems presented in Section 4 encourage us to challenge the more complicated system. In this section, two different types of fuzzy inferences will be discussed. The simulation results will also be compared.

Suppose we use n data points to form the basic GM(1, 1) model [1]. Since every prediction model will introduce a certain degree of predicted error, a compensation system is thus necessary to narrow the

predicted error. In [7], a type-1 fuzzy rule [10] is considered as a candidate of the compensation system and is formulated as follows:

If x_1 is A_{1i} and x_2 is A_{2i} and...and x_n is A_{ni}, then y is c.

In this rule, x_1 to x_n correspond to the data points used in constructing the grey model [1] and A_{1i} to A_{ni} are the linguistic descriptors. The consequent part y, a constant, is the predicted error resulting from the grey modeling. The gradient descent method [6–9] is often exploited to adjust the membership functions. As we know, the tuned result from the gradient descent method is heavily dependent on the initial settings and the learning rates. Besides, before performing the simulation, we have to take the derivatives of the defined error function with respect to every parameter involved. Different shapes of membership functions defined will also result in re-taking the derivatives. To avoid doing this kind of cumbersome work, we employ the genetic algorithms to tune the membership functions.

To further reduce the total number of fuzzy rules in the rule base, we take the difference between two consecutive data as a single variable [7]. Therefore, if we use four data points to construct the GM(1, 1) model, instead of using four variables, only three are needed in the antecedent part. This will greatly alleviate the load in tuning the desired parameters. For example, if each variable in the antecedent part is assigned five labels, then a total of 125 rules exists in the rule base since the rules are automatically generated from the defined labels in each variable. Under the same condition, if each datum, not the difference between two consecutive data, is considered as a single variable, a total of 625 rules comes out. Besides, 15 more parameters are needed to be determined by the genetic algorithms due to the added variable. This is because three parameters are required to characterize a nonsymmetric triangular membership function. This may in turn result in an increase of 515 parameters to be tuned by the genetic algorithms. Besides, treating the difference between two consecutive data as a single variable allows us to narrow the searching space in finding the optimum solutions. Of course, using four variables in the antecedent part may have different effects on the final results. From the viewpoint of lowering the number of parameters to be tuned by the genetic algorithms, however, we prefer to use only three variables in the premise.

To better relate the output from the system with the system inputs, we also consider another commonly referred inference type, i.e., type-3 fuzzy rule [10]. The type-3 fuzzy rule can be described as follows:

If x_1 is A_{1i} and x_2 is A_{2i} and...and x_n is A_{ni}, then y is f.

In this case, the consequent part in the fuzzy rule is a function of the inputs. That is, the consequence is a linear combination of the x_i's. In our case, $f = p_0 + p_1x_1 + p_2x_2 + p_3x_3$ since only three variables are used in the premise. The coefficients p_i's are to be determined by the genetic algorithms. For the type-3 fuzzy rules, if each input variable is assigned five labels, a total of 500 p_i's have to be determined from the generated 125 rules. Besides, each input membership function is characterized by three parameters, a total of 45 parameters resulting from those three input variables also needs to be tuned. As a result, the genetic algorithms must be able to simultaneously adjust those 545 parameters during the learning phase. Again, if four variables are used in the antecedent part, an increase of 2015 parameters is imposed on the genetic algorithms.

6. Simulation results and discussion

In this section we will present the simulation results obtained from the proposed fuzzy grey models. In applying the genetic algorithms to the fuzzy grey system, we set (population size, P_c, P_m) = (30, 0.6, 0.0000654) and (population size, P_c, P_m) = (30, 0.6, 0.0000204) for the genetic algorithms to optimize the underlying type-1 and type-3 fuzzy rules, respectively. Besides, each case evolves for 300 generations. To test the effectiveness of the proposed algorithms, we forecast the monthly average temperatures for the city of St. Louis, Missouri based on the data given in [11]. The data collected from January 1955 to December 1974 are depicted in Fig. 5. The first 18 years' data are treated as the training patterns and the last two years' as the test ones. Besides, to expedite the optimizing process without sacrificing the overall performance, each fuzzy model selects only

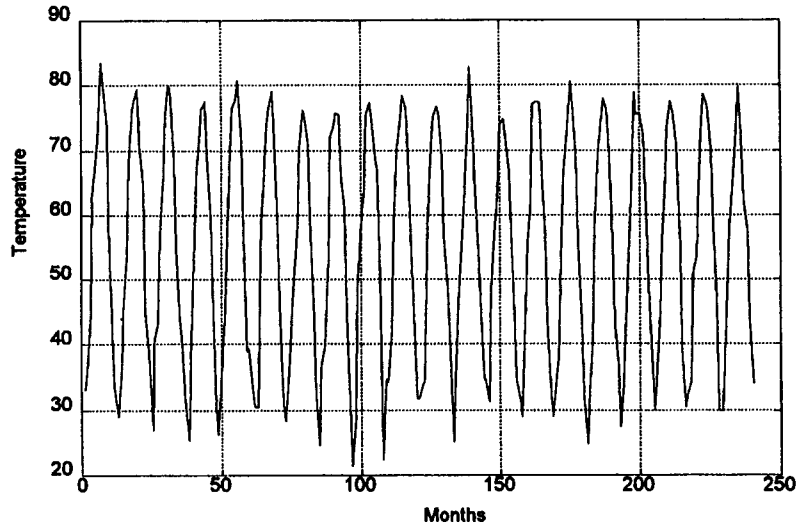

Fig. 5. The monthly average temperatures (in °F) for the city of St. Louis, Missouri, from January 1955 to December 1974.

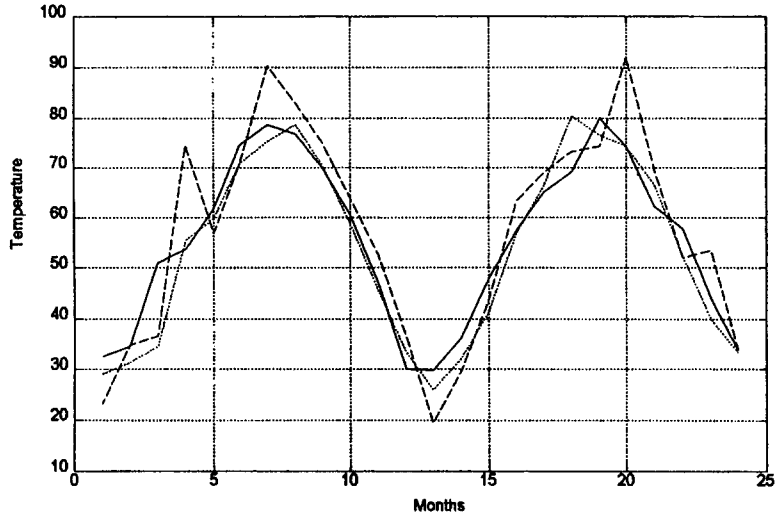

Fig. 6. The forecast outputs between January 1973 and December 1974 for St. Louis by the type-1 fuzzy rules. In this figure, the solid, dashed, and dotted curves represent the outputs from the actual, the GM(1, 1), and our models, respectively.

one-third of the past patterns as sample data. Having analyzed with the grey relational method described in Section 2, the chosen patterns are the ones which show stronger similarity to the current test pattern. We use four data points to form the GM(1, 1) model such that the temperature of the following month can be easily predicted. The type-1 fuzzy model is then used to decide the quantity of compensation for the predicted output from the grey model. Fig. 6 shows the simulation results from both uncompensated and compensated grey models. The compensated curve can be seen to resemble more closely the actual one than does the grey model.

Another commonly used type of fuzzy model, i.e., type-3, is also adopted to perform the compensation job. Simulation results from different conditions assigned for genetic algorithms in type-3 fuzzy grey models are shown in Figs. 7 and 8. Note that the

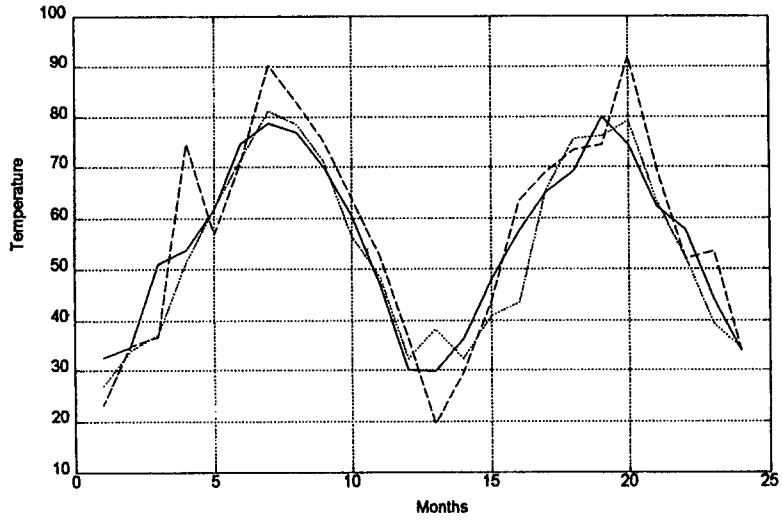

Fig. 7. The forecast outputs (based on 7-bit coding scheme) between January 1973 and December 1974 for St. Louis by the type-3 fuzzy rules. In this figure, the solid, dashed, and dotted curves represent the outputs from the actual, the GM(1, 1), and our models, respectively.

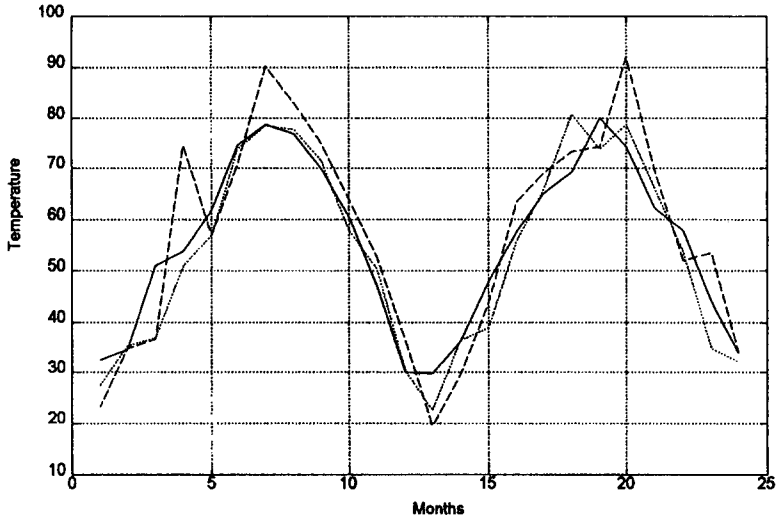

Fig. 8. The forecast outputs (based on 9-bit coding scheme) between January 1973 and December 1974 for St. Louis by the type-3 fuzzy rules. In this figure, the solid, dashed, and dotted curves represent the outputs from the actual, the GM(1, 1), and our models, respectively.

results plotted in Fig. 7 are obtained by using the 7-bit coding scheme whenever a parameter is selected to perform the crossover or mutation operation. While in Fig. 8, the 9-bit coding scheme, which is also the case for other simulations presented in this paper, is used. Empirically, a higher precision coding scheme will normally result in a better performance. Although different lengths of binary coding may result in different degrees of compensation, the presented models did demonstrate their capabilities in lowering the predicted errors caused by the original grey model.

We then perform the same prediction work for the city of Taipei, Taiwan, based on the data given in [7]. The training data range from January 1974 to December 1991. The tests data consist of the years of 1992 and 1993. Besides, we use four data points to form the

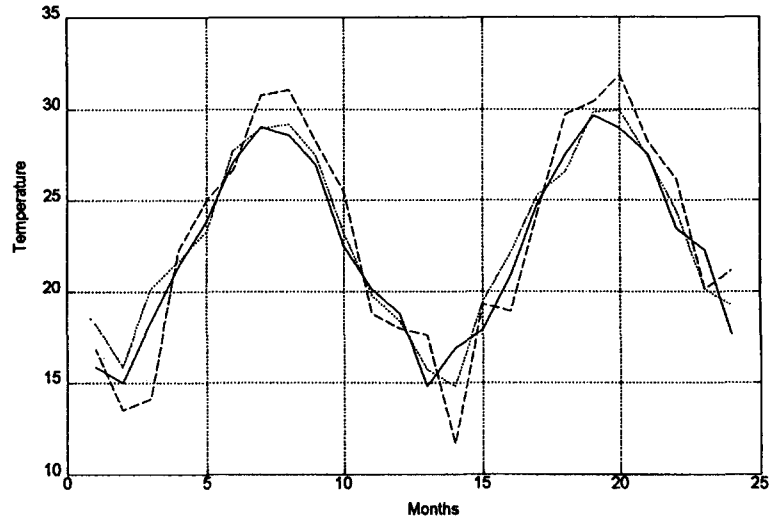

Fig. 9. The forecast outputs between January 1992 and December 1993 for the city of Taipei by the type-1 fuzzy rules. In this figure, the solid, dashed, and dotted curves represent the outputs from the actual, the GM(1, 1), and our models, respectively.

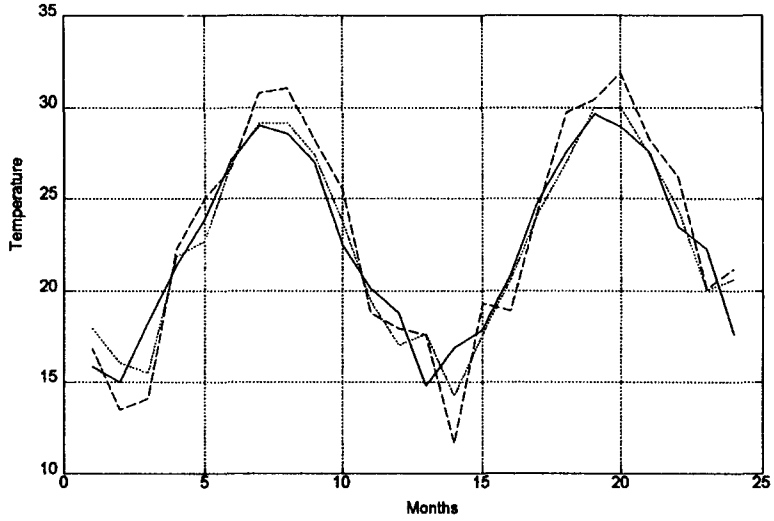

Fig. 10. The forecast outputs between January 1992 and December 1993 for the city of Taipei by the type-3 fuzzy rules. In this figure, the solid, dashed, and dotted curves represent the outputs from the actual, the GM(1, 1), and our models, respectively.

basic GM(1, 1) model. The prediction results from both type-1 and type-3 fuzzy grey models are shown in Figs. 9 and 10, respectively. Again, the presented compensation models outperform the grey model (Table 1). It is worthy of notice that the results from type-1 fuzzy grey model more closely resemble the actual ones than does type-3 model. This may be due to the fact that more parameters are needed to be optimized in type-3 model than in type-1.

As we know, different population sizes, crossover rates, or mutation rates selected for the genetic algorithms may have different effects on the final outcome. The work presented here is not to emphasize how to find the optimal parameters for the fuzzy model;

Table 1
The comparisons between the prediction errors from different types of fuzzy models and grey system

	St. Louis, Missouri	Taipei, Taiwan
Type-1 fuzzy model (%)	7.289323	4.668775
Type-3 fuzzy model (%)	8.032210	6.016611
Grey model (%)	13.967464	9.416272

instead, we focus on the fact that the genetic algorithms can be easily applied to the fuzzy grey model to perform the compensation job. In this paper, type-2 fuzzy model, i.e., the consequence is also a fuzzy set, is not considered. This is because no *ad hoc* designed fuzzy rule base is known in advance. Unless we also apply genetic algorithms or some other techniques to establish the rule base, there is no way to decide the consequent parameters by using the genetic algorithms. We believe that if a well-defined rule base can be constructed in advance, the performance based on type-2 model should be comparable to the results shown here. This is based on the fact that fewer parameters are required to be adjusted by the genetic algorithms.

7. Conclusion

The fuzzy inference model is integrated into the grey model to compensate for the predicted outputs from the grey system. Two well-known types of fuzzy models, i.e., type-1 and type-3, are adopted to perform the compensation job. Since each roughly determined fuzzy model needs to be adjusted to perform a satisfactory job, the proposed real-valued genetic algorithms are applied to the optimization of the necessary parameters needed in the fuzzy model. Unlike most of the conventional genetic approaches which can only deal with the binary encoding of parameters, the presented work can directly handle parameters expressed in the form of real values. The binary coding procedure is required only when a specific parameter is selected to perform the crossover or mutation operation. Simulation results from predicting the monthly average temperatures for two different cities are shown to verify that the integrated genetic algorithms and the fuzzy model can make up an effective compensation model for the grey system. The most difficult problem encountered in this work is that no prior information is known about the exact range of each to-be-determined parameter in the consequent part of the fuzzy rule. As a result, there is little chance to find the optimum set of consequent parameters. Although the searching ranges of parameters are unknown, the presented results still demonstrate the applicability of the genetic algorithms to the fuzzy grey system. Future work can focus on how to decide the range for each consequent parameter such that the performance can be further improved.

References

[1] J.L. Deng, Introduction to grey system theory, *J. Grey System* **1**(1) (1989) 1–24.
[2] J.L. Deng, Properties of multivariable grey model GM(1,N), *J. Grey System* **1**(1) (1989) 25–41.
[3] D.B. Fogel, An introduction to simulated evolutionary optimization, *IEEE Trans. Neural Networks* **5**(1) (1994) 3–14.
[4] D.E. Goldberg, *Genetic Algorithms in Search, Optimization & Machine Learning* (Addison-Wesley, Reading, MA, 1989).
[5] J.J. Grefenstette, Optimization of control parameters for genetic algorithms, *IEEE Trans. System, Man Cybernet.* **16**(1) (1986) 122–128.
[6] F. Guely and P. Siarry, Gradient descent method for optimizing various fuzzy rule bases, in: *Proc. FUZZ-IEEE* (1993) 1241–1246.
[7] Y.-P. Huang and C.-C. Huang, The integration and application of fuzzy and grey modeling methods, *Fuzzy Sets and Systems* **78** (1996) 107–119.
[8] Y.-P. Huang and C.-C. Huang, The applications of fuzzy and gradient descent methods to the grey prediction control, *J. Grey System* **7**(1) (1995) 9–22.
[9] H. Nomura, I. Hayashi and N. Wakami, A learning method of fuzzy inference rules by descent method, in: *Proc. FUZZ-IEEE* (1992) 203–210.
[10] T. Takagi and M. Sugeno, Fuzzy identification of systems and its applications to modeling and control, *IEEE Trans. System, Man Cybernet.* **15**(1) (1985) 116–132.
[11] L.X. Wang and J.M. Mendel, Generating fuzzy rules from numerical data, with applications, USC-SIPI Report #169, Univ. of Southern California, Los Angeles, CA (1991).
[12] L.A. Zadeh, Fuzzy sets, *Inform. and Control* **8** (1965) 338–353.