Identifying Temporal Pattern Clusters to Predict Events in Time Series

Heesoo Hwang

Abstract - This paper proposes a method for identifying temporal pattern clusters to predict events in time series. Instead of predicting future values of the time series, the proposed method forecasts specific events that may be arbitrarily defined by the user. The prediction is defined by an event characterization function, which is the target of prediction. The events are predicted when the time series belong to temporal pattern clusters. To identify the optimal temporal pattern clusters, fuzzy goal programming is formulated to combine multiple objectives and solved by an adaptive differential evolution technique that can overcome the sensitivity problem of control parameters in conventional differential evolution. To evaluate the prediction method, five test examples are considered. The adaptive differential evolution is also tested for twelve optimization problems.
Keywords - event prediction, supervised clustering, fuzzy goal programming, differential evolution

1. Introduction

Time series analysis is fundamental to engineering and scientific endeavors. In many real applications, studying the change of temporal features of a non-stationary time series and identifying the features that are representing the significance of time instances are important. Traditional time series analysis employs statistical methods to model and predict future values of the time series. Traditional time series analysis methods, such as the Box-Jenkins method, are limited by the stationarity of the time series and the normality and independence of the residuals [7] - [9]. However, for most real time series, those conditions are not satisfied. One of the most severe drawbacks of the approach is its inability to identify complex characteristics, which are due to characterizing and predicting all points in a time series.

Data mining is the exploration of data with the goal of discovering hidden patterns. Its uniqueness is found in the problems addressed-those with large data sets and complex, hidden relationships. The approaches for times series data mining require prior knowledge of the types of structures or temporal patterns to be discovered and represent these temporal patterns as a set of templates. The use of predefined templates prevents achieving the basic data mining goal of uncovering useful, novel, and hidden temporal patterns [10] - [12]. More recent works for time series data mining focus on overcoming the limitations of conventional time series data mining methods [4,5].

This paper proposes a new approach to applying data mining concepts to time series analysis, which establishes a method of uncovering hidden patterns in time series data. The method focuses on characterizing and predicting events, important occurrences localized in time. The method is capable of handling non-stationary, non-periodic, irregular, and chaotic deterministic time series. The prediction model is composed of temporal pattern clusters optimized by adaptive differential evolution(ADE) which is capable of adapting control parameters of conventional differential evolution(DE). The robustness of the proposed ADE is tested for twelve representative optimization problems. The event prediction method is also evaluated for five test examples.

Manuscript received: Jan. 22, 2002 accepted: July 31, 2002
Heesoo Hwang is with the School of Electrical Engineering at Halla University, San 66 Hungup-ri, Hungup-myun, Wonju-si, 220-712 Kangwon-do, Korea.

2. Event Prediction

This section explains the concepts of events, cluster models for event prediction, supervised clustering, and the combination of multiple objectives.

2.1 Cluster model for event prediction

In a time series, events are important occurrences. An event is defined depending on the prediction goal. For example, sharp rises or falls of a stock price are defined as events. Let X and Z be the daily open prices of stock to be used in the modeling of event prediction and its evaluation, respectively. The stock will be bought at the open of the first day and sold at the open of the second day. The goal is to pick buy and sell days that will have greater than expected increases. Thus, the events are those days when the price increases more than 5%, defining the event characterization function $g(t)$, which must clearly classify events and non-events in data.

Now let data satisfying an event characterization function in X be X_{event} and others be X^c_{event}. A temporal pattern is a hidden structure in a time series that is characteristic and predictive of events. The purpose of the event clustering is to uncover optimal hidden temporal pattern clusters detecting $g(X_{event})$ from X. The identified clusters are used to predict events $g(Z_{event})$ in the testing data Z. Event clustering is based on the assumption that the observations of similar characteristics in time series form adjacent points in multi-dimensional space.

Let's consider data as shown in Fig. 1, in which · represents an event X_{event} and x represents a non-event X^c_{event}. The number of optimal temporal pattern clusters needed to predict events, namely, to classify X_{event} alone in X, is one; that is self-evident. The temporal pattern cluster is a ellipsoid centered on a cluster centroid with radius r, which has a value in the range of $[r_{\min}, r_{\max}]$. The use of $r_{\min}$ as a value of r makes it not only less likely to classify non-events as events, but also more likely to classify events as non-events. On the other hand, $r_{\max}$ makes it not only more likely to classify non-events as events, but also less likely to classify events as non-events. The value of r has a fuzzy feature in the range of $[r_{\min}, r_{\max}]$; it is uncertain how data of similar characteristics are located adjacently in multi-dimensional space. To deal with this, we introduce the fuzzy membership function of Eq.(1).

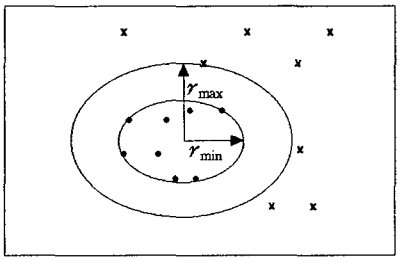


Fig. 1 Temporal pattern cluster model for event prediction

Data located inside the cluster with radius $r_{\min}$ have membership value 1, indicating the crisp event, and data located outside the cluster with radius $r_{\max}$ have membership value 0, indicating the crisp non-event. On the other hand, data located both outside the cluster with radius $r_{\min}$ and inside the cluster with radius $r_{\max}$ have a membership value in the range of [0,1], which is proportional to the distance as in Eq.(1).

$$u_i = \begin{cases} 0, & d_{ik} > r_{imax} \\ 1, & d_{ik} \le r_{imin} \\ 1-(d_{ik}-r_{imin})/(r_{imax}-r_{imin}), & r_{imin} < d_{ik} < r_{imax} \end{cases} \quad (1)$$

Here d_{ik} is the distance between the kth data and the ith temporal pattern cluster centroid as in Eq.(2). u_i is membership value of the kth data belonging to the ith temporal pattern cluster.

$$d_{ik} = \sqrt{a_{1i}\cdot(x_{1k}-c_{1i})^2 + \cdots + a_{di}\cdot(x_{dk}-c_{di})^2} \quad (2)$$

Here x_{jk} is the jth component of the kth data with d dimension and c_{ji} is the jth component of the ith cluster center. a_{ji} is the weighting factors for the jth dimension of the ith cluster. During the evolutionary clustering, the weighting factors are searched in the range of real values. The negative weighting factors are considered to be zero, so the corresponding dimensions are neglected in the calculation of Eq.(2). If all remaining a_{ji}s have the same values, then the cluster is a hyper-sphere; otherwise, it is an ellipsoid. The ith temporal pattern cluster is defined by its centroid, radii r_{imin} and r_{imax}, and its weighting factors.

If more than one clusters is identified, we utilize the concept of fuzzy decision making to determine the degree of an event. The maximum of the membership values of Eq.(1) for the clusters is taken as in Eq.(3). If the maximum value is higher than a threshold δ, then the event is predicted. As the threshold value, usually 0.5 is taken.

$$u = \max(u_{ij}, i = 1, \cdots, c) \quad (3)$$

c is the number of clusters, and u_{ij} is the membership value of the jth data for the ith cluster.

The fuzzy membership functions will contribute to classifying the event and non-event data adjacently located in multi-dimensional space.

2.2 Supervised clustering

An event prediction model is the result of supervised clustering. Clustering is used to partition the data into a number of clusters based on their distance measures. Two issues have been commonly encountered when applying clustering; the determination of the number of clusters in the data and the ability of the algorithm to find clusters containing a highly varying number of data objects. A method that is applied often in the literature is repetitive clustering with various numbers of clusters and partition assessment with cluster validity index [13]. This approach is time consuming since it requires repetitive modeling. Moreover, the nature of the data is such that the validity measures typically give no insight into suitable partitioning because no distinctive local minima of the validity function is usually found [14]. To overcome this problem, supervised event clustering based on multiple objectives is proposed. As the clustering progresses, a multi-objective guided search automatically determines

suitable partitions.

The event clustering is achieved by not only the maximization of event classification defined by Eq.(4) but also the minimization of overlapped clusters and cluster size defined by Eqs.(5) and (6), respectively. After the clustering, useless clusters that contain no data or only overlapped data are deleted and minimum and maximum radii of the clusters are calculated without deteriorating the performance of the event classification.

$$f_{event}=\frac{1}{n}\cdot\left(\sum_{i=1}^{nX_{event}}B_i+\sum_{i=1}^{nX^c_{event}}B_i^c\right) \tag{4}$$

Here n is the number of multi-dimensional data. nX_{event} is the number of events and nX^c_{event} the number of non-events, so $n=nX_{event}+nX^c_{event}$. B_i is 1 if the ith datum, which is an event, is classified as an event; otherwise, it is 0. B_i^c is 1 if the t ih datum, which is a non-event, is classified as a non-event; otherwise, it is 0.

$$f_{area}=\frac{1}{n}\sum_{i=1}^{n}M_i \tag{5}$$

Here n is the number of multi-dimensional data, and M_i is the frequency of the ith data belonging to different clusters.

$$f_{radius}=\sum_{i=1}^{c}r_i \tag{6}$$

Here c is the number of clusters, and r_i is the radius of the ith cluster.

Eqs(4), (5) and (6) have different objectives, that must be combined into one, which will be achieved by fuzzy goal programming.

2.3 Combination of multiple objectives

Optimization is an important activity in many areas of science and engineering. The classical framework for the optimization is the minimization (or maximization) of the objectives given the constraints for the problem to be solved. However, many design problems are characterized by multiple objectives, and a trade-off among various objectives must be made, leading to under or over-achievement of different objectives. The fuzzy logic approach formulates the objectives and the constraints as membership functions that represent the degree that each objective is satisfied on a scale of [0,1]. Since Zimmermann introduced the concept of fuzzy set in optimization, various fuzzy optimization methods have been proposed including fuzzy goal programming and genetic algorithms for the programming [15] - [17].

In event clustering, objectives with different ranges of goals are scaled to the range of [0,1] through membership functions. To do this, we adopted Eq.(7) for the maximization of Eq.(4) and Eq.(8) for the minimization of Eqs.(5) and (6), respectively.

$$\mu(f_i)=\begin{bmatrix}0, & f_i< f_{i-\min}\\ 1, & f_i> f_{i-\max}\\ \dfrac{f_i-f_{i-\min}}{f_{i-\min}-f_{i-\max}}, & f_{i-\min}<f_i<f_{i-\max}\end{bmatrix} \tag{7}$$

Here f_i is the ith objective function value, $\mu(f_i)$ the membership value of f_i, $f_{i-\min}$ the minimum value of f_i, and $f_{i-\max}$ the maximum value of f_i.

$$\mu(f_i)=\begin{bmatrix}1, & f_i< f_{i-\min}\\ 0, & f_i> f_{i-\max}\\ \dfrac{f_{i-\min}-f_i}{f_{i-\min}-f_{i-\max}}, & f_{i-\min}<f_i<f_{i-\max}\end{bmatrix} \tag{8}$$

Here f_i is the ith objective function value, $\mu(f_i)$ the membership value of f_i, $f_{i-\min}$ the minimum value of f_i, and $f_{i-\max}$ the maximum value of f_i.

In event clustering, $f_{i-\min}$ and $f_{i-\max}$ are set to 0 and 1 for f_{event} and to 1 and 0 for f_{area} and f_{radius}, respectively. By the weighted sum of the membership values of the objectives, we combine the multiple objectives into one goal function as in Eq.(9). The trade-off among the objectives can be realized by changing the associated weight factors. The weight factors have values in the range of [0,1] and influence the relative importance of the objectives.

$$\mu_{total}(f)=\beta_{event}\mu(f_{event})+\beta_{area}\mu(f_{area})+\beta_{radius}\mu(f_{radius}) \tag{9}$$

Here $\mu_{total}(f)$ is the final goal function to maximize, The weight factor β_{event} is set to 1, and β_{area} and β_{radius} are set to 0.01.

3. Optimization

To solve the fuzzy goal programming previously described, DE is utilized. DE evolves a randomly selected population that is initialized with a larger number of individuals (temporal pattern clusters) than assumed necessary. As the evolution progress, radii of clusters that don't contain event data become 0s and overlapped regions among clusters become smaller. When the termination condition of the evolution is met, any redundant and useless clusters that don't contain any event or unique events of their own are deleted from the best temporal pattern clusters. For the remaining clusters, the minimum and maximum radii of each cluster are calculated. The c_{ji}s, r_{imin}s, r_{imax}s, a_{ji}s of the remaining clusters constitute the model to predict events.

3.1 Differential evolution

DE is a very simple but powerful population-based, stochastic function minimizer. DE turned out to be best genetic algorithm for solving the real-valued test function suite of the first ICEO [1,2]. DE is a parallel direct search method, that utilizes parameter vectors as a population for each generation. The initial population is chosen randomly. The crucial idea behind DE is a scheme for generating trial parameter vectors. DE generates new parameter vectors by adding the weighted difference vector between population members to another member. If the resulting vector yields a lower objective function value than a predetermined population member, the newly generated vector replaces the vector with which it was compared.

3.1.1 The DEk variant

We tried a variant of DE that works as follows: for each vector $x_{i,G}(\ i=1,\cdots,np)$, a trial vector $v_{i,G+1}$ is generated by Eq.(10) which is called DEk to differentiate between variants of DE.

$$v_{i,G+1}=\frac{x_{best,G}+x_{r1,G}}{2}+F\cdot(x_{r2,G}+x_{r3,G}-x_{r4,G}-x_{r5,G})$$
$$\text{with}\quad r_1,\ r_2,\ r_3,\ r_4,\ r_5\in[1,np] \tag{10}$$

$x_{best,G}$ is the best performing vector until the current generation. $x_{r1,G}$, $x_{r2,G}$, $x_{r3,G}$, $x_{r4,G}$, and $x_{r5,G}$ are mutually different vectors selected randomly in the population. F is a positive real and constant factor that controls the amplification of the differential variation $(x_{r2,G}+x_{r3,G}-x_{r4,G}-x_{r5,G})$. G is generation, and np is population size. To increase the potential diversity of the perturbed parameter vectors, crossover is introduced. Let's consider a vector $x'_{i,G+1}$ with d dimension of Eq.(11). A crossover operation of Eq.(12) combines $x_{i,G}$ and $v_{i,G+1}$ and results in Eq.(11).

$$x'_{i,G+1}=(x_{1i,G+1},x_{2i,G+1},\ldots,x_{di,G+1}) \tag{11}$$

$$x'_{ji,G+1}=(v_{ji,G+1}*x_{ji,G})\quad j=1,\ldots,d \tag{12}$$

$*$ is the crossover operator and herein means uniform crossover of Eq.(13).

$$x'_{ji,G+1}=\begin{bmatrix} v_{ji,G+1} & \text{if}\,(rand\le Cr) \\ x_{ji,G} & otherwise \end{bmatrix} \tag{13}$$

$rand\in[0,1]$ is a random variable and Cr is the crossover rate.

To decide whether or not it should become a member of generation $G+1$, the new vector $x'_{i,G+1}$ of Eq.(11) is compared to $x_{i,G}$. If $x'_{i,G+1}$ yields a smaller objective function value than $x_{i,G}$, then $x_{i,G+1}$ is set to $x'_{i,G+1}$; otherwise, the old value $x_{i,G}$ is retained.

3.1.2 Sensitivity of control parameters

To check the performance of DEk, we selected twelve optimization test functions considering continuous and discontinuous, convex and non-convex, single modal and multi-modal, low dimensional and high dimensional, and deterministic and stochastic properties as shown in the appendix. When the global minimum was 0, we defined the minimization task to be completed once the final value was obtained with an accuracy better than 10-6. For f_4, we chose a value less than 15 to indicate the global minimum and a value less than 0.998004 in case of f_5. We experimented with ten test runs with randomly chosen initial parameter vectors for each test function and each minimization and measured the average number of function evaluations required to find the global minimum. To find optimal parameters Cr and F of DEk, we executed the experimentation at points with Cr and F 0.2 to 1.0 with a 0.1 increase step. Fig. 2 and Fig. 3 show the average number of function f_1 evaluations with different population sizes.

According to the graphs, the average number of function evaluations is very sensitive to population size, F, and Cr. As the population increases, the algorithms are more likely to converge but provide slower convergence. This is common in evolutionary algorithms and population size is usually set to 30 to 100 depending on the complexity of the problems to solve. However, the values of F and Cr can't be so easily determined. Genetic algorithms(GAs) have been widely studied, so the effective values of crossover rate and mutation rate are known, but that is not the case with DE. Therefore, we need a mechanism to adapt the control parameters of DEk depending on the problems.

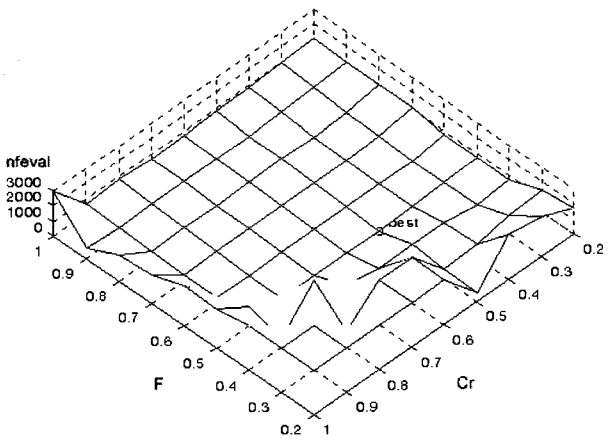


Fig. 2 Average number of f_1 evaluations depending on F and Cr (population size = 6)

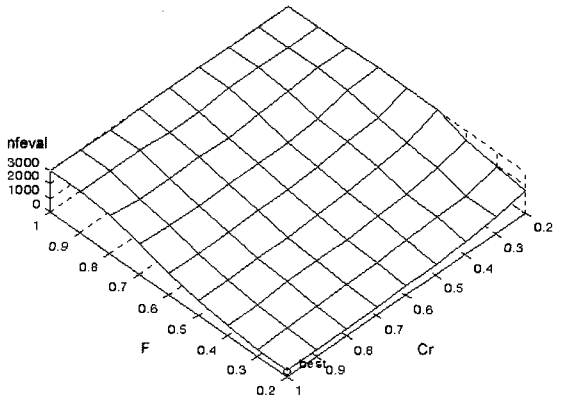


Fig. 3 Average number of f_1 evaluations depending on F and Cr (population size = 30)

3.2 Adaptation of control parameters

To overcome the sensitivity of the control parameters, we propose a self-adaptive mechanism for DEk. The self-adaptation eliminates difficulty in selecting effective control parameters for specific problems. As a result, it reinforces the robustness of DEk and facilitates its use. Basically, the adaptive mechanism is based on roullet-wheel selection. For every generation, a control parameter is selected in a predefined parameter set and the probability of the control parameter being selected is proportional to the amount that objective function value is improved by the control parameter. Regulating the numbers of the improvement is especially important in a small parameter set. In the early stage of evolution, a few control parameters in the set are extraordinary. The parameters take over a significant proportion, other parameters lack a fair opportunity of contributing to improve the objective function value. The linear scaling of the selected numbers will help. Now let's consider adaptive differential evolution.

3.2.1 Adaptive F DEk

DEk with the adaptation of control parameter F is refferred to as AFDEk. A summary of the AFDEk procedure follows.

Step 1: Initialize the vector population composed of np individual vectors with d dimension generated randomly: $P(t)=\{a_1(t), a_2(t), \cdots, a_{np}(t)\}$, $a_k=\{x_{1k}, \cdots, x_{dk}\}$. Define the parameter set $F_s=\{F_1, \cdots, F_{mk}\}$ for F. Define $nF_s=\{nF_1, \cdots, nF_{mk}\}$ to record the selected numbers of each element in F_s and to initialize to 1s. In this paper, we used $F_s=\{0.2, 0.25, 0.3, \cdots, 1.0\}$, mk = 17.

Step 2: For every population vector, evaluate the objective function ($Func$): $\Phi(t)=\{\Phi(a_1(t)), \cdots, \Phi(a_{np}(t))\}$,

$$\Phi(a_k(t))=Func(x_{1k}, \cdots, x_{dk}).$$

Step 3: For each individual vector $a_k(t)(\ i=1, \cdots, np)$, select a value for control parameter F as follows:

$$p_k=\sum_{j=1}^{i}\frac{nF_j}{\sum_{l=1}^{mk}nF_l},\quad \text{if}\ (p_{i-1}<rand\leq p_i)\ F=F_i.$$

$rand\in[0,1]$ is a randomly generated value. Select five mutually different vectors a_{r1}, a_{r2}, a_{r3}, a_{r4}, and a_{r5} in the current population, make a trial vector, and then do a crossover as follows:

$$v_k(t)=\frac{a_{best,G}+a_{r1,G}}{2}+F\cdot(a_{r2,G}+a_{r3,G}-a_{r4,G}-a_{r5,G})$$
$$x'_k(t)=v_k(t)*a_k(t).$$

Step 4: For every population vector, evaluate the objective function: $\Phi(t)=\{\Phi(x'_1(t)), \cdots, \Phi(x'_{np}(t))\}$. If the goal is to maximize the objective function, if $(\Phi(x'_k(t))>\Phi(a_k(t)))$, then $a_k(t)=x'_k(t)$ and $nF_i=nF_i+1$.

Step 5: To prevent the overuse of a specific F_i, the nF_i is adjusted as follows: $const$=3.0, $nF_{\min}$, $nF_{\max}$, and nF_{avg} are the minimum, maximum, and mean values of nF_s, respectively.

```
For i=1 to mk {
    if (nF_min > (const · nF_avg − nF_max)
△ = nF_max − nF_avg {
        ; a=(const−1) · nF_avg/△;
          b=nF_avg · (nF_max − const · nF_avg)/△; }
    else {
        △ = nF_avg − nF_min; a=nF_avg/△;
        b=− nF_min · nF_avg/△; }
    nF_i=a · nF_i+b;
}
```

Step 6: Check the termination condition. If it is satisfied, stop. Otherwise, go to step 3.

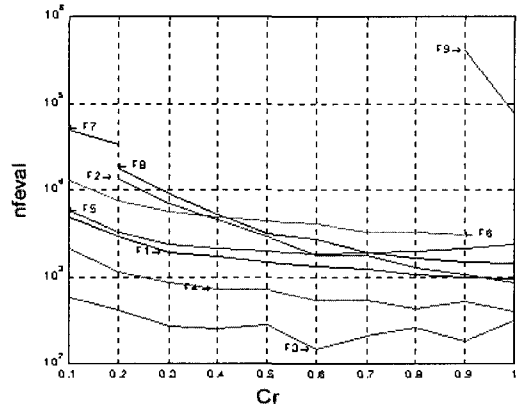


Fig. 4 Average number of function evaluations by AFDEk

For Cr 0.1 to 1 with steps of 0.1 we experimented with AFDEk for inne test functions $f_1 \sim f_9$. For each function and each parameter value of Cr, we tested ten runs and averaged the number of function evaluations. The results are shown in Fig. 4. Except for f_7 and f_9, the average number of function evaluations is not sensitive to Cr in the range between 0.5 and 0.9. However, f_7 and f_9 require extremely different values of Cr, so Cr must be adapted.

3.2.2 Adaptive F and Cr DEk

DEk with the adaptation of control parameters F and Cr in the range between 0.2 and 1.0 is called AFCDEk. The adaptation mechanism for Cr is identical to that described for F. C_{rs} and nC_{rs} are defined, and the same initial values as used for F_s and nF_s are used again. We experimented with AFCDEk for nine test functions $f_1 \sim f_9$. For each function, we tested ten runs and averaged the number of function evaluations. The results with the best control parameters are summarized in Table 1. For AFCDEk, the only control parameter to consider is population size.

In Table 1, x indicates that global minimum could be not found, n.a. stands for "not applicable," * indicates that not all of the test runs provided the global minimum, ANM is the annealed version of the Nelder & Mead strategy, ASA is adaptive simulated annealing, and DE1 and DE2 are two different DE schemes [3]. For those problems where ANM or ASA could find the minimum, DEs usually converged faster, especially in the more difficult cases. Since DE is inherently parallel, a further speed up can be implemented by parallel computation. AFDEk and AFCDEk show promising results. The performance of AFDEk is superior to that of DEs in most cases. AFCDEk converged a little slower than AFDEk, but its deterioration is less severe except the case of $f1$. This case requires an extremely low value of Cr as shown in Fig. 4, which seems to cause the failure to find the global minimum in all the test runs. According to Table 1, DEs could find the global optimum with a comparatively small population size. The small population usually contributes to fast convergence, so AFDEk, AFCDEk, and other evolutionary algorithms must be compared with the same population size.

Using test sets of three objective functions, f_{10}, f_{11} and f_{12}, an experimental comparison of the algorithms was performed. The test functions are representatives of the classes of uni-modal, multi-modal, and discontinuous functions. The population size for the algorithms is 200. For f_{11}, 40,000 function evaluations were performed for each run, and for f_{10} and f_{12}, the number is increased to 100,000. Twenty runs per algorithm are carried out. Table 2 summarizes the mean and standard deviations of the

Table 1 Comparison of test results for function optimization

	ANM	ASA	DE1	DE2	AFDEk	AFCDEk
	T TF NV nfeval	TRS TAS nfeval	NP F Cr nfeval	NP F Cr nfeval	NP Cr nfeval	NP nfeval
f1	0 n.a. 1 95	1E-5 10 397	10 0.5 0.3 490	6 0.95 0.5 392	10 0.9 253	6 318
f2	0 n.a. 1 106	1E-5 1E5 11275	6 0.95 0.5 746	6 0.95 0.5 615	10 1 275	15 1098
f3	300, 0.998 20 90258	1E-7 100 354	10 0.8 0.3 915	20 0.95 0.2 1300	20 0.8 882	10 1048
f4	300 0.98 30 x	1E-5 100 4812	10 0.75 0.5 2378	10 0.95 0.2 2873	10 0.8 1945	10 2651
f5	3000 0.995 50 X	1E-5 100 1379	15 0.9 0.3 735	20 0.95 0.2 828	15 0.6 1086	15 1155
f6	5E6 0.995 50 x	1E-5 100 3581	10 0.4 0.2 834	10 0.9 0.2 1125	12 0.8 1499	12 1763
f7	10 0.99 50 x	1E-5 0.1 x	30 1 0.3 22167	20 0.99 0.2 12804	20 0.2 25085	30 *
f8	5 0.95 5 2116	1E-6 300 11864	10 0.8 0.5 1559	10 0.9 0.9 1076	12 0.9 786	12 1018
f9	5E4 0.995 150 x	1E-8 700 x	100 0.65 1 165680	80 0.6 1 254824	80 1 51761	80 165380

(NP: population size, nfeval: number of function evaluations)

Table 2 Comparison of AFDEk, AFCDEk, and other evolutionary algorithms

		ES1	ES30	Meta-EP	GA	AFDEk	AFCDEk
		NP 20 Nsigma 1	NP 200 Nsigma 30	NP 200	NP 200 Cr 0.6 Mr 0.001	NP 200	NP 200
f11	avg	1.08E-5	6.672E-1	1.998E2	1.65E2	1.09E-8 Cr: 0.7	3.4E-6
	std	4.73E-6	2.610E-1	6.564E1	4.74E1	2.61E-8	6.9E-6
f12	avg	4.1	0	0	5.39E1	0 all Crs	0
	std	3.177	0	0	4.74E1	0	0
f10	avg	1.326	1.618E-3	1.976	5.253	0.62E-13 Cr: 0.6	5.8E-12
	std	1.039	9.290E-4	6.300E-1	5.1E-1	2.0E-13	9.4E-12

(NP: population size, avg: average number of function evaluations, std: standard deviation)

best objective function values within the last generation of the runs.

The parameters used in Evolution Strategy (ES1, ES30), Meta-Evolutionary Programming (Meta-EP), and GA are defined in the literature [6]. The comparison of just three objective functions prevents drawing general conclusions, but Table 1 and Table 2 show that AFDEk and AFCDEk perform very well. Thus other advantages are easiness of use and robustness; population size is the only parameter considered.

4. Test Examples

To evaluate the proposed event prediction method, five test examples are considered. For all cases, the initial parameter settings are as follows: cluster number is 10, population size is 50, generation number is 500.

4.1 Nonlinear system

In this section, a static and nonlinear system of Eq.(14) is considered. Fifty input-output data are obtained [18]. The data of x_3 and x_4 are given as dummy inputs to check the modeling method's the ability to uncover hidden structures. The first half of the data is utilized for building up an event prediction model and the rest for the evaluation of the model. As an event characterization function, Eq.(15) is used.

$$y=(1+x_1^{-2}+x_2^{-1.5})^{2}, \quad 1\le x_1, x_2\le 5 \tag{14}$$

$$g(y\ge 2.5) \tag{15}$$

As a result, three temporal pattern clusters for the events are obtained as in Fig. 5. The result shows the weighting factors a_{ji} of x_3 and x_4 in Eq.(2) are all zeros, i.e.. considering x_3 and x_4 is unnecessary when we predict events since x_3 and x_4 are dummy inputs. Fig. 6 shows the prediction result for the evaluation data. Considering that an outlier (data not made by Eq.(14)) is present in the evaluation data, the prediction is excellent.

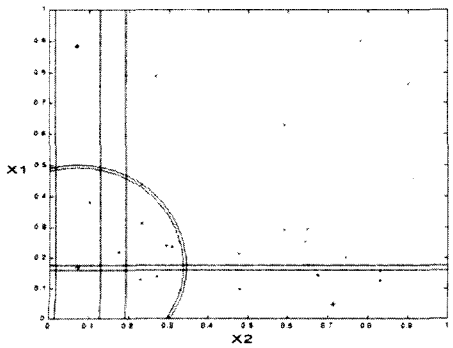


Fig. 5 Event prediction model (prediction accuracy=100%)

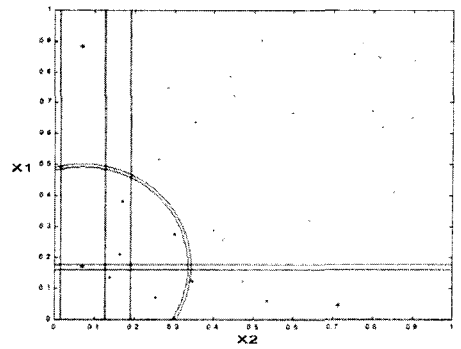


Fig. 6 Result of event prediction (prediction accuracy=96%)

4.2 Box and Jenkins' gas furnace

As a dynamical process, we consider a gas furnace with single input $u(t)$ (gas flow rate) and single output $y(t)$ (CO_2 concentration). Among 296 data points, the first half of the data is utilized for building up an event prediction model and the rest for the evaluation of the model [8,18]. As an event characterization function, Eq.(16) is used. Under the assumption, we don't know input-output relationships to describe the process effectively, we consider ten variables; $y(t-1)\cdots y(t-4)$, $u(t-1)\cdots u(t-6)$, as inputs. Of course, we know two inputs, and $u(t-4)$ and $y(t-1)$ are usually enough to define the process.

$$g(50\le y(t)\le 57) \tag{16}$$

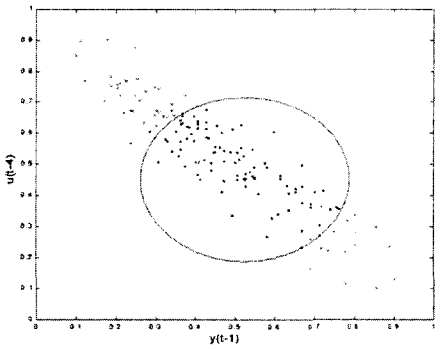


Fig. 7 Event prediction model (prediction accuracy=98.6%)

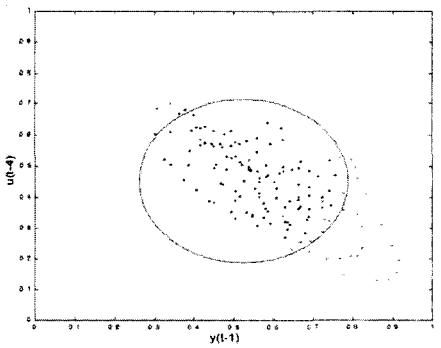


Fig. 8 Result of event prediction (prediction accuracy=93.1%)

As a result, one temporal pattern cluster is identified as in Fig. 7, in which the only two inputs, $u(t-4)$ and $y(t-1)$ contribute to the cluster, which means that the method is capable of discovering hidden patterns in data. Fig. 8 shows the prediction result for the evaluation data.

4.3 Chaotic Mackey-Glass time series

As an example of chaotic process, we consider a Mackey-Glass time series that is quasi-periodic and chaotic [19]. From Eq.(17), 1500 data points are generated. 500 data points are used for building an event prediction model and the next 500 data points for the evaluation of the model. Inputs are $x(t)$, $x(t-6)$, $x(t-12)$, and $x(t-18)$, and output is $x(t+6)$.

$$\dot{x}(t) = \beta x(t) + \frac{\alpha x(t-\tau)}{1+x^{10}(t-\tau)} \tag{17}$$

where $\alpha=0.2$, $\beta=-0.1$, $\tau=17$, $x(0)=1.2$.

The results of the prediction are shown in Table 3 for five different event functions characterized by Eq.(18). The average prediction accuracy for the modeling and evaluation is more than 95.6% and 94.7%, respectively.

$$g_x = \frac{x(t+6)-x(t)}{x(t)} \tag{18}$$

Table 3 Results of event prediction in chaotic time series

	$g(g_x>0.05)$	$g(0.01<g_x\leq 0.05)$	$g(0\leq g_x<0.01)$	$g(-0.01\leq g_x<0)$	$g(g_x<-0.01))$
No. of clusters	1	2	2	2	2
Prediction accuracy in modeling	100%	96.47%	92.4%	90.2%	98.8%
Prediction accuracy in evaluation	99.6%	95.6%	90.8%	89.2%	98.2%

4.4 Trend of stock price

As an example of financial data, we deal with stock price data, which consists of 100 data points, ten inputs, $x_1(t)\cdots x_{10}(t)$, and one output, $y(t)$. A detailed description of the variables is explained in the literature [18]. The results of the prediction are shown in Table 4 for five different event characterization functions. The average prediction accuracy is more than 92.8%.

Table 4 Results of event prediction for stock price

	$g(y\geq 5)$	$g(5>y\geq 0)$	$g(1>y>-1)$	$g(0>y\geq -10)$	$g(y<-10)$
No. of clusters	2	1	2	1	1
Prediction accuracy	91%	87%	94%	93%	99%

4.5 Boston housing price

This example uses housing price data in Boston [20]. Instead of trying to predict all housing values, we try to predict the value of the top 20% of houses, a task similar to that often carried out in market research. We consider 506 cases in which each case describes eleven house characteristics that might be expected to affect house prices. 350 cases are used for building the prediction model and the rest are used for the evaluation. The event prediction aims to forecast the value of the top 20% of houses. Table 5 shows the comparative result with See5 [20], which has a reputation for fast and robust classification algorithms. The predictions show reliable performance.

Table 5 Comparative results of event prediction for Boston housing values

Algorithm	Prediction accuracy	
	in modeling	in evaluation
Our method	96.28%	96.15%
See5	97.71%	93.59%

5. Conclusion

In this paper a method for identifying temporal pattern clusters to predict events in a time series was proposed. To identify optimal temporal pattern clusters, fuzzy goal programming is formulated to combine multiple objectives and solved by ADE. The presented ADE technique tackled the sensitivity problem of control parameters in conventional DE. The test results for twelve optimization problems showed ADE had the best performance compared with other evolutionary algorithms. The event prediction method was evaluated for five test examples and yielded meaningful results of a prediction accuracy of more than 93%. Even with a complex, non-stationary, chaotic time series, such as a gas furnace, Mackey-Glass data, and stock price, the method uncovered predictive hidden patterns. For the top 20% of Boston house values, our method resulted in a comparatively reliable prediction model.

Reference

[1] K. Price and R. Storn, "Differential Evolution: Numerical Optimization Made Easy," Dr. Dobbs Journal, pp 18 - 24, April 1997.

[2] R. Storn, "Minimizing the Real Functions of the ICEC96 Contest by Differential Evolution," IEEE Conference on Evolutionary Computation, pp. 842 - 844, Nagoya, 1996.

[3] R. Storn and K. Price, Differential Evolution - a Simple and Efficient Adaptive Scheme for Global Optimization over Continuous Spaces, Technical Report TR-95-012, International Computer Science

Institute, University of California at Berkeley, 1995.
[4] R. J. Povinelli. "Using Genetic Algorithms to Find Temporal Patterns Indicative of Time Series Event s," Data Mining with Evolutionary Algorithms Workshop, Genetic and Evolutionary Computation Conference, pp. 80 - 84, 2000.
[5] R. J. Povinelli and X. Feng. "Characterization And Prediction Of Welding Droplet Release Using Time Series Data Mining," Artificial Neural Networks in Engineering Conference, pp. 857 - 862, 2000.
[6] T. Back and H.-P. Schwefel, "An Overview of Evolutionary Algorithms for Parameter Optimization," Evolutionary Computation, vol. 1, no. 1, pp. 1 - 23, 1993.
[7] S. M. Pandit and S.-M. Wu, Time Series and System Analysis with Applications, Wiley, 1983.
[8] G. E. P. Box and G. M. Jenkins, Time Series Analysis: Forecasting and Control, Holden-Day, 1976.
[9] B. L. Bowermann and R. T. O'Connell, Forecasting and Time Series: An Applied Approach, Duxbury Press, 1993.
[10] D. J. Berndt and J. Clifford, "Finding Patterns in Time Series: A Dynamic Programming Approach," Advances in Knowledge Discovery and Data Mining, AAAI Press, pp. 229 - 248, 1996.
[11] E. J. Keogh and M. J. Pazzani, "An Enhanced Representation of Time Series which Allows Fast and Accurate Classification, Clustering and Relevance Feedback," Proc. of AAAI Workshop on Predicting the Future: AI Approaches to Time Series Analysis, Madison, Wisconsin, 1998.
[12] M. T. Rosenstein and P. R. Cohen, "Continuous Categories for a Mobile Robot," Proc. of Sixteenth National Conference on Artificial Intelligence, 1999.
[13] X. L. Xie and G. A. Beni, "Validity Measure for Fuzzy Clustering," IEEE Trans. on Pattern Anal. Machine Intell., vol. 3, pp. 841 - 846, 1991.
[14] M. Setnes and U. Kaymak, "Fuzzy Modeling of Client Preference from Large Data Sets: An Application to Target Selection in Direct Marketin g," IEEE Trans. on Fuzzy Systems, vol. 9, no. 1, pp. 153 - 163, February, 2001.
[15] R. Narasimhan, "Goal Programming in a Fuzzy Environment," Decision Sciences, vol. 11, pp. 325 - 336, 1980.
[16] R. N. Tiwari, S. Dhamar and J. R. Rao, "Fuzzy Goal Programming-An Additive Model," Fuzzy Sets and Systems, vol. 24, pp. 27 - 34, 1987.
[17] M. Gen and B. Liu, "A genetic algorithm for nonlinear goal programming," Research Report, vol. 23, pp. 141 - 148, Ashikaga Institute of Technology, 1996.
[18] M. Sugeno and T. Yasukawa, "A Fuzzy Logic Based Approach to Qualitative Modeling," IEEE Trans. on Fuzzy Systems, vol. 1, no. 1, pp. 7 - 31, Feb. 1993.
[19] S. L. Chiu, "Fuzzy Model Identification Based on Cluster Estimation," Journal of Intelligent and Fuzzy Systems, vol. 2, no. 3, 1994.
[20] See5, RuleQuest, http://www.rulequest.com

Appendix

Twelve optimization problems selected to test adaptive differential follows.

$f_1 = \sum_{i=1}^{3} x_i^2$, $-5.12 \le x_i \le 5.12$; the minimum is 0.

$f_2 = 100 \cdot (x_1^2 - x_2)^2 + (1 - x_1)^2$,
$-2.048 \le x_i \le 2.048$; the minimum is 0.

$f_3 = 25 + \sum_{i=1}^{5} \lfloor x_i \rfloor$; $\lfloor x_i \rfloor$ rounds x_i to the nearest integer. $-5.12 \le x_i \le 5.12$; the minimum is 0.

$f_4 = \sum_{i=1}^{30} (i \cdot x_i^4 + rand())$, $-1.28 \le x_i \le 1.28$; the minimum is 0. *rand*() is a random variable with uniform distribution.

$$f_5 = \frac{1}{0.002 + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^{2} (x_i - a_{ji})^6}},$$

$-65.536 \le x_i \le 65.536$, and the minimum is 0.998004 where $a_{j1} = -32, -16, 0, 16, 32$, $j = 0, 1, 2, 3, 4$, $a_{j1} = a_{j(\mathrm{mod})5,0}$, and mod means remainder after division. $a_{j2} = a_{j+k} (k = 1, 2, 3, 4)$ a_j2 $a_{j2} = -32, -16, 0, 16, 32$, $j = 0, 5, 10, 15, 20$, and .

$$f_6 = \sum_{i=1}^{4} \begin{cases} 0.15(z_i - 0.05 sgn(z_i))^2, & |x_i - z_i| < 0.05 \\ d_i \cdot x_i^2, & |x_i - z_i| \ge 0.05 \end{cases}$$

where $z_i = \lfloor \left| \frac{x_i}{0.2} \right| + 0.49999 \rfloor \cdot sgn(x_i) \cdot 0.2$, $d_i = 1, 1000, 10, 100$, $|x_i| < 0.05$ $i = 1, 2, 3, 4$ $-1000 \le x_i \le 1000$, and the minimum is 0.

$f_7 = \sum_{i=1}^{10} \frac{x_i^2}{4000} - \Pi_{10}^{i=1} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$, $-400 \le x_i \le 400$; the minimum is 0.

$f_8 = 9 - x_1 - x_2$, $x_1 > 0$ $x_2 > 0$, $(x_1 - 3)^2 + (x_2 - 2)^2 \le 16$, $x_1 x_2 \le 14$. The minimum is 0.

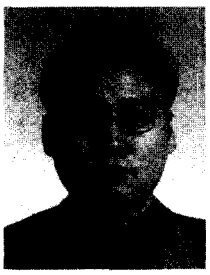
$f_9 = \sum_{i=0}^{16} x_i \cdot z^i$ If $z_i \in [-1, 1]$, then $f_9(x_i, z_i) \in [-1, 1]$, else if $z_i = \pm 1.2$, then $f_9(x_i, z_i) \ge T_{16}(1.2)$. $-1000 \le x_i \le 1000$; the minimum is $T_{16}(z_i) = 10558.1450229$.

$f_{10} = -20 \exp\left[-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{30} x_i^2}\right] - \exp\left[\frac{1}{n}\sum_{i=1}^{30} \cos(2\Pi x_i)\right]$

$+20+e$, $-30 \le x_i \le 30$; the minimum is 0.

$f_{11} = \sum_{i=1}^{30} x_i^2$, $-30 \le x_i \le 30$; the minimum is 0.

$f_{12} = \sum_{i=1}^{30} \lfloor x_i + 0.5 \rfloor^2$, $\lfloor x_i + 0.5 \rfloor$ rounds $x_i + 0.5$ to the nearest integer, $-30 \le x_i \le 30$; the minimum is 0.



Heesoo Hwang received his B.S., M.S. and Ph.D. degrees in Electrical Engineering from Yonsei University in 1986, 1988, and 1993, respectively. From 1993 to 2000, he served as a senior and a principal engineer for the Korea High-Speed Rail Development project in Korea Railroad Research Institute and Korea High Speed Railway Construction Authority, respectively. Since 2001, he has been an assistant professor in the School of Electrical Engineering at Halla University. His research interests include pattern classification, event prediction, and surveillance using fuzzy logic, neural networks, and evolutionary algorithms.