An Evolutionary Approach to Pattern-Based Time Series Segmentation

Fu-Lai Chung, *Member, IEEE*, Tak-Chung Fu, Vincent Ng, and Robert W. P. Luk, *Senior Member, IEEE*

***Abstract*—Time series data, due to their numerical and continuous nature, are difficult to process, analyze, and mine. However, these tasks become easier when the data can be transformed into meaningful symbols. Most recent works on time series only address how to identify a given pattern from a time series and do not consider the problem of identifying a suitable set of time points for segmenting the time series in accordance with a given set of pattern templates (e.g., a set of technical patterns for stock analysis). However, the use of fixed-length segmentation is an oversimplified approach to this problem; hence, a dynamic approach (with high controllability) is preferable so that the time series can be segmented flexibly and effectively according to the needs of the users and the applications. In view of the fact that this segmentation problem is an optimization problem and evolutionary computation is an appropriate tool to solve it, we propose an evolutionary time series segmentation algorithm. This approach allows a sizeable set of pattern templates to be generated for mining or query. In addition, defining similarity between time series (or time series segments) is of fundamental importance in fitness computation. By identifying the perceptually important points directly from the time domain, time series segments and templates of different lengths can be compared and intuitive pattern matching can be carried out in an effective and efficient manner. Encouraging experimental results are reported from tests that segment both artificial time series generated from the combinations of pattern templates and the time series of selected Hong Kong stocks.**

***Index Terms*—Data mining, evolutionary algorithms, stock time series, time series segmentation.**

I. Introduction

RECENTLY, the increasing use of temporal data has initiated various research and development efforts in the field of data mining. Time series are an important class of temporal data objects and can be easily obtained from financial and scientific applications (e.g., daily temperatures, weekly sales totals, and prices of mutual funds and stocks). They are, in fact, major sources of temporal databases and, undoubtedly, discovering useful time series patterns is of fundamental importance. Unlike transactional databases with discrete items, time series data are characterized by their numerical and continuous nature. Reference [1] suggests dividing the sequences into meaningful subsequences and representing them using real-valued functions. Furthermore, there is a need to discretize a continuous time series into meaningful labels/symbols [2], [3]. We call this process "numeric-to-symbolic" (N/S) conversion, and consider it to be one of the most fundamental components in time series data mining systems.

In [2], a simple N/S conversion method is proposed. A fixed length window is used to segment a time series into subsequences and a time series is then represented by the primitive shape patterns that are formed. This discretization process mainly depends on the choice of the window width. However, using fixed-length segmentation is an oversimplified approach to solve the problem. There are at least two identified disadvantages. First, with fixed-length subsequences, only patterns whose length does not vary will be considered in the mining process. However, meaningful patterns typically appear with different lengths throughout a time series. Second, as a result of the even segmentation of a time series, meaningful patterns may be missed if they are split across time points. Thus, a dynamic approach is better, which identifies the time points in a more flexible way [4] (i.e., using different window widths), but this is certainly not a trivial segmentation problem.

In [5] and [6], it is suggested that we identify the time points at which behavior changes occur in a time series. In the statistics literature, this is called the "change-point detection problem." The standard solution involves fixing the number of change points beforehand, identifying their positions, and then determining functions for curve fitting the intervals between successive change points. In [7], an iterative algorithm is proposed that fits a model to a time segment and then uses a likelihood criterion to determine if the segment should be partitioned further. In [8], it is suggested to discover the underlying switching process in a time series, which entails identifying the number of subprocesses and the dynamics of each subprocess. The nonlinear gated experts concept from statistical physics was proposed to perform the segmentation. In [9] and [10], dynamic programming is proposed to determine the total number of intervals within the data, the location of these intervals, and the order of the model within each segment. In [11], the segmentation problem is considered with a new tool for exploratory data analysis and data mining, called the scale-sensitive gated experts (SSGE), which can partition a complex nonlinear regression surface into a set of simpler surfaces (called "features").

The segmentation problem has also been considered from the perspective of finding cyclic periodicity for all of the segments. In [12] and [13], the data cube and the *a priori* data mining techniques were used to mine segment-wise periodicity using a fixed length period. An offline technique for the competitive identification of piecewise stationary time series is described in [14]. In addition to performing piecewise segmentation and

Manuscript received January 15, 2003; revised March 1, 2004. This work was supported in by the UGC CERG under Project PolyU 5065/98E.

The authors are with the Department of Computing, Hong Kong Polytechnic University, Hunghom, Kowloon, Hong Kong (e-mail: cskchung@comp.polyu.edu.hk; cstcfu@comp.polyu.edu.hk; cstyng@comp.polyu.edu.hk; csrluk@comp.polyu.edu.hk).

Digital Object Identifier 10.1109/TEVC.2004.832863

identification, the proposed technique maps similar segments of a time series as neighbors on a neighborhood map.

As we can see, much of the recent research on this and similar problems can be characterized procedurally in the following general manner [15]: 1) find an approximation and robust representation for a time series, for example, Fourier coefficients, piecewise linear models, etc.; 2) define a flexible matching function that can handle various pattern variations (scaling, transformations, etc.); and 3) provide an efficient scalable algorithm, using the adopted representation and matching function, for massive time series data sets. Although these approaches can generally identify a given pattern from a time series, they do not consider the problem of identifying a suitable set of time points in a time series when a set of pattern templates is given; for example, the technical patterns (e.g., head-and-shoulder, double top, etc.) for stock analysis. Further, in order to form a versatile mining space, a variety of patterns (e.g., in different resolutions) have to be identified. The aforementioned segmentation task can be regarded as an optimization problem, where evolutionary computation can help. In this paper, an evolutionary time series segmentation algorithm for stock data mining is proposed. With respect to the target application, for fitness evaluation, we propose a perceptually important point (PIP)-based subsequence-matching scheme of which a preliminary version was presented in [16]. In addition, parameters are introduced to control the identification of subsequence patterns with different characteristics (i.e., in different temporal and amplitude scales). The paper is organized into six sections. In Section II, the PIP-based pattern-matching scheme for fitness evaluation is described. The proposed segmentation algorithm is introduced in Section III. Section IV describes the methods used to control the evolutionary time series segmentation process. The simulation results are reported in Section V and Section VI provides the conclusions of the paper.

II. Flexible Pattern Matching Scheme for Fitness Evaluation

To facilitate evolutionary time series segmentation, defining a distance measure or the similarity between time series (or time series segments) for fitness evaluation is of fundamental importance. In [16], we proposed a flexible time series pattern-matching scheme that was based on the fact that interesting and frequently appearing patterns are typically characterized by a few critical points. For example, the head-and-shoulder pattern consists of a head point, two shoulder points, and a pair of neck points. These points are perceptually important in the human visual identification process; therefore, they should similarly be taken into account in the pattern matching process. The proposed scheme follows this idea by locating the perceptually important points (PIPs) in a data sequence $P = \{P_k | k = 1, \ldots, m\}$, where m denotes the length of P, in accordance with a query sequence $Q = \{Q_k | k = 1, \ldots, n\}$, where n denotes the length of Q. The location process works as follows.

A. Identification of Perceptually Important Points

With sequences P and Q being normalized to a unit interval, i.e., maximum and minimum of P_k (and Q_k) equal to 1 and 0,

```
Function PIPlocate (P,Q)
    Input: sequence P[1..m], length of Q[1..n]
    Output: pattern SP[1..n]
Begin
    Set SP[1]=P[1], SP[n]=P[m]
    Repeat until SP[1..n] all filled {
      Select point P[j] with maximum distance
      to the adjacent points in SP
      Add P[j] TO SP }
    Return SP
End
```

Fig. 1. Pseudocode of the PIP identification process.

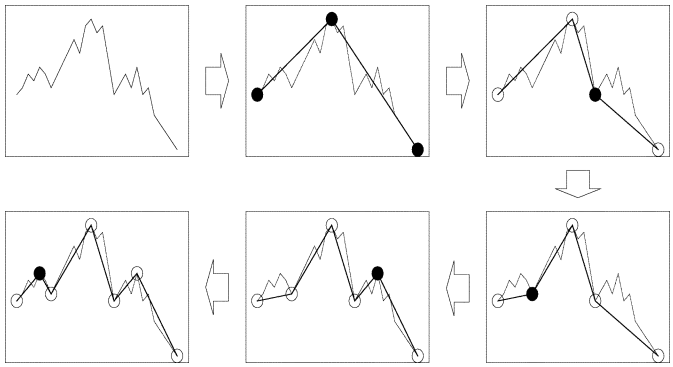

Fig. 2. Identification of seven PIP's (head-and-shoulder pattern).

respectively, the PIPs are located in order according to the pseudocode described in Fig. 1. The first two PIPs that are found will be the first and last points of P. The next PIP that is found will be the point in P with maximum distance to the first two PIPs. The fourth PIP that is found will then be the point in P with maximum distance to its two adjacent PIPs, either in-between the first and second PIPs or in-between the second and the last PIPs. The process of locating the PIPs continues until the number is equal to the length of the query sequence Q. To determine the maximum distance to the two adjacent PIPs $p_1(x_1, y_1)$ and $p_2(x_2, y_2)$, we calculate the vertical distance between a test point $p_3(x_3, y_3)$ and a line connecting the two adjacent PIPs, that is

$$\begin{aligned} VD(p_3, p_c) &= |y_c - y_3| \\ &= \left| \left(y_1 + (y_2 - y_1) \cdot \frac{x_c - x_1}{x_2 - x_1} \right) - y_3 \right| \end{aligned} \quad (1)$$

where $p_c(x_c, y_c)$ is the point with $x_c = x_3$ on the line connecting p_1 and p_2. Such a function is intended to capture the fluctuation in the sequence and those highly fluctuating points are considered as PIPs.

To illustrate the identification process, the "head-and-shoulder" pattern is used. Fig. 2 shows the result when the number of data points in the input sequence P is 29 and the query sequence Q is 7 (i.e., $m = 29$ and $n = 7$).

B. Distance Measure

After identifying the PIPs in the data sequence, the amplitude distance (AD) between the data sequence P and the query

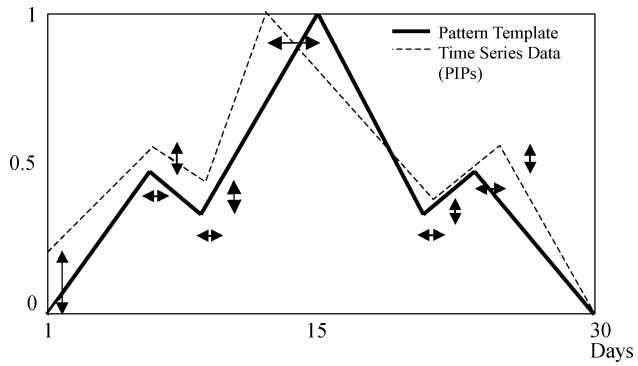

Fig. 3. Amplitude distance measures (up-down arrows) and temporal distance measures (left-right arrows).

sequence Q can be computed using direct point-to-point comparison, i.e.,

$$AD(P,Q) = \sqrt{\frac{1}{n}\sum_{k=1}^{n}(SP_k - Q_k)^2} \tag{2}$$

where $SP = \{SP_k | k = 1, \ldots, n\}$ denotes the PIPs found in P. However, the measure in (2) has not yet taken the horizontal scale (time dimension) into consideration. So, it is preferable to consider the horizontal distortion of the pattern against the pattern templates. The temporal distance (TD) between P and Q is defined as follows:

$$TD(P,Q) = \sqrt{\frac{1}{n-1}\sum_{k=2}^{n}(SP_k^t - Q_k^t)^2} \tag{3}$$

where SP_k^t and Q_k^t denote the time coordinate of the PIP SP_k extracted from P and the time coordinate of the query sequence point Q_k, respectively. The amplitude and temporal distance measures are illustrated in Fig. 3. To take into consideration both horizontal and vertical distortion in our fitness evaluation, the distance (or similarity) measure could be modified as follows:

$$DM(SP,Q) = w_1 \times AD(SP,Q) + (1 - w_1) \times TD(SP,Q) \tag{4}$$

where w_1 denotes the weighting for the amplitude distance and the temporal distance and this can be specified by the users. According to our experiments, 0.4 is a reasonable choice for this setting.

III. Evolutionary Time Series Segmentation

Obviously, static N/S conversion is only a basic conversion approach. A dynamic approach can be derived if the time points can be identified in an irregular way. This is an optimization problem where evolutionary computation can contribute [17], [18]. In this section, we propose an evolutionary time series segmentation algorithm for stock price data. A set of pattern templates $\{Q\}$, particularly, the technical patterns, is assumed given for fitness evaluation purposes. The evolutionary segmentation process can be described as follows.

```
Initialization of population U(1)
For t = 1 to MaxGen
   Evaluate the fitness of each individual
    Ci in U(t) with the given set of
    pattern templates {Q}
   If min(fitness(Ci)) < MinFitness then
      Stop
   Else
      Apply the following steps to produce
         the next generation U(t+1):
         - Selection
         - Crossover & Mutation
   End If
End For
```

Fig. 4. Overall evolutionary segmentation process.

A. Chromosome Representation

Unlike the traditional genetic algorithms (GAs) that adopt binary bit strings to encode a chromosome, a more direct representation is used in our approach. The time points identified in each individual will form the genes themselves, for example, $C_a = \{4, 9, 14, 16, 19\}$ represents a time series being cut at time points 4, 9, 14, 16, and 19. In other words, the first segment lasts from data point 1 to data point 4, the second segment starts at data point 5 and ends at data point 9, and so on. Note here that the variable length of individuals (chromosomes) can be represented and this representation is the same as that used in [19] but the problem domains are completely different.

B. Algorithmic Framework

Initially, a set of user-defined parameters has to be set as follows.

PopSize: Size of the population.
MaxGen: Maximum number of generations.
MinFitness: Minimum fitness value.
dlen: Desired segment length.

The evolutionary segmentation process is executed according to the pseudocode given in Fig. 4. The fittest individual C_{fittest} with the best fitness value in the last generation will be the final segmentation result.

C. Initialization of the Population

The first step of the evolutionary segmentation process is to initialize a set of time points (the genes) for the chromosomes in the first generation. Here, we have adopted a user-oriented approach, where users specify the desired segment length $dlen$ during the evolutionary process. It is a reasonable assumption that users can specify whether they want to investigate a long-term or a short-term period/pattern. Moreover, it is only an approximate parameter for the initialization process. With this parameter, the approximate number of time points (ds) for a chromosome is given by

$$ds = \frac{L}{dlen} \tag{5}$$

where L is the length of the whole time series. The initial population `U(1)` can be generated as described in the pseudocode of Fig. 5. Using such a scheme, the initial time points will be distributed quite evenly within the chromosome.

For Each individual C_i in $U(1)$ (where i = 1 to $PopSize$)
  Evenly distribute the set of time points $(D_1^i,\ldots,D_{n_i}^i)$ where n_i=ds
  For Each D_j^i in P_i (where j = 1 to n_i)
    Randomly move it between D_{j-1}^i and D_{j+1}^i
  End For
  The time series s is segmented as
  $s=[s(1)\ldots s(D_1^i),s(D_1^i+1)\ldots s(D_2^i),\ldots,s(D_{n_i}^i+1)\ldots s(L)]$
End For

Fig. 5. Initialization process of population $U(1)$.

Crossover Procedure

1. Pick two individuals C_a and C_b from the pool of selected parents.
2. Randomly generate a crossover-point (*CutPt*).
3. Divide C_a into time point sequences (new genes) $(D_1^a,\ldots,D_x^a)$ and $(D_{x+1}^a,\ldots,D_{n_a}^a)$, where $x \le CutPt$.
4. Divide C_b into time point sequences (new genes) $(D_1^b,\ldots,D_y^b)$ and $(D_{y+1}^b,\ldots,D_{n_b}^b)$, where $y \le CutPt$.
5. Combine the two sets of new genes to form offspring, i.e. new individuals
   $C_{new1} = (D_1^a,\ldots,D_x^a,D_{y+1}^b,\ldots,D_{n_b}^b)$
   and
   $C_{new2} = (D_1^b,\ldots,D_y^b,D_{x+1}^a,\ldots,D_{n_a}^a)$.

Fig. 6. Crossover operation in evolutionary segmentation process.

D. Selection

Selection in evolutionary algorithms is aimed at giving a higher probability for reproduction to fitter individuals in a population so that their favorable characteristics can be inherited by even fitter offspring. This is where the principle of "survival of the fittest" applies. Here, we keep the best parent (based on the fitness value) from the current population $U(t)$ as one of the candidates in the next generation. For the other candidates, the roulette selection scheme [19] is applied. The individuals selected will then go through crossover and mutation.

E. Crossover

The essence of any crossover operator is to exchange the components of two parents to form new offspring. In our experiments, it was found that a crossover rate $R_c = 0.6$, or higher, produced good results. Single-point crossover is realized by cutting the chromosomes at a randomly chosen position and then swapping the segments between the two parents. We formulate the single-point crossover in our algorithm as shown in Fig. 6.

For example, if we have two individual C_a and C_b

$$C_a = \{4, 6, 9, 14, 16, 20\}$$
$$C_b = \{2, 7, 12, 16, 19\}$$

and (the crossover-point) $CutPt = 10$. The new individuals (offspring) will then be

$$C_{\text{new1}} = \{4, 6, 9\} U \{12, 16, 19\} = \{4, 6, 9, 12, 16, 19\}$$

Mutation Procedure

1. Set the user-defined parameters
   R_a: probability of adding a time point during mutation
   R_d: probability of dropping a time point during mutation is set as 1 - R_a
2. Pick an individual C_i from the pool of selected parents for mutation.
3. Add or drop a time point in C_i. That is, for a random number N_r uniformly generated in [0,1], a time point is added if $N_r \le R_a$; otherwise a time point is dropped.

Fig. 7. Mutation operation in evolutionary segmentation process.

and

$$C_{\text{new2}} = \{2, 7\} U \{14, 16, 20\} = \{2, 7, 14, 16, 20\}.$$

F. Mutation

Mutation in evolutionary algorithms is another search operator. Its main function is to introduce new genetic material and maintain a certain level of diversity in a population since crossover does not introduce any new genetic material. In our approach, the remaining candidates of the next generation (after crossover) are formed by the mutation operations whose procedure is described in Fig. 7.

G. Fitness Evaluation

With a set of pattern templates $\{Q\}$, the proposed PIP-based similarity measure (from Section II) can be adopted to evaluate

```
For Each individual C_i in U(t)
  For Each segment C_{i,j} in C_i
  Extract the PIP's from C_{i,j} and store them in SP
    fitness(C_{i,j}) = min_{∀Q}{DM(SP,Q)}
  End For
    fitness(C_i) = 1/(n_i+1) Σ_{j=1}^{n_i+1} fitness(C_{i,j})
End For
```

Fig. 8. Fitness evaluation in evolutionary segmentation process.

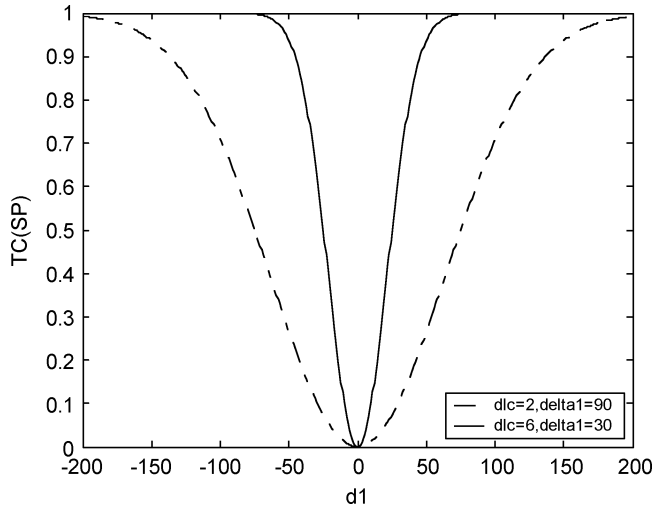

Fig. 9. Temporal control penalty function $(dlen = 180)$.

the fitness of each subsequence segmented by the identified time points. Thus, the lower of the fitness value, the better the corresponding chromosome. The fitness of the individuals in a population can be computed as described in the pseudocode of Fig. 8. Note here that the smaller the distance between the segment $C_{i,j}$ and a template Q from the set of templates $\{Q\}$, the lower the fitness value and, consequently, the better the corresponding chromosome (segmentation solution).

IV. Segmentation With High Controllability

As mentioned in Section III-C, users may prefer to investigate a specific resolution for patterns in a time series (i.e., long-term or short-term). Furthermore, in stock data, price fluctuation analysis is also very important; hence, the patterns (segments) of interest may be in different amplitude scales. For stock data mining, creating a huge pattern space with a wide variety of patterns (e.g., the coexistence of long-term and short-term patterns) is very important. Thus, we propose the introduction of controllability in both the temporal and amplitude domains of our evolutionary segmentation scheme. By setting different parameters, users can have more control over the desired time series segmentation results.

A. Temporal Control

In the temporal domain, the length of the segments (and also the number of time points) is uncontrollable once we start the evolutionary (iterative) process of the proposed segmentation scheme. Therefore, we prefer a mechanism that can control the length of the segments toward the length specified by the users during the evolutionary process. To achieve this goal, a temporal control penalty function is proposed and defined as follows:

$$TC(SP) = 1 - \exp^{-\left(\frac{d_1}{\theta_1}\right)^2} \tag{6}$$

where $d_1 = slen - dlen$, that is, the difference between segment length $(slen)$ and the desired segment length $(dlen)$ specified by the users. The parameter θ_1 is used to control the sharpness of the function, hence, the strength of the temporal control. It is defined as follows:

$$\theta_1 = \frac{dlen}{dlc} \tag{7}$$

where dlc is the desired length control parameter. Larger dlc values will lead to smaller θ_1 values and this will strengthen the temporal control (i.e., a segment length closer to the desired length is greatly preferred).

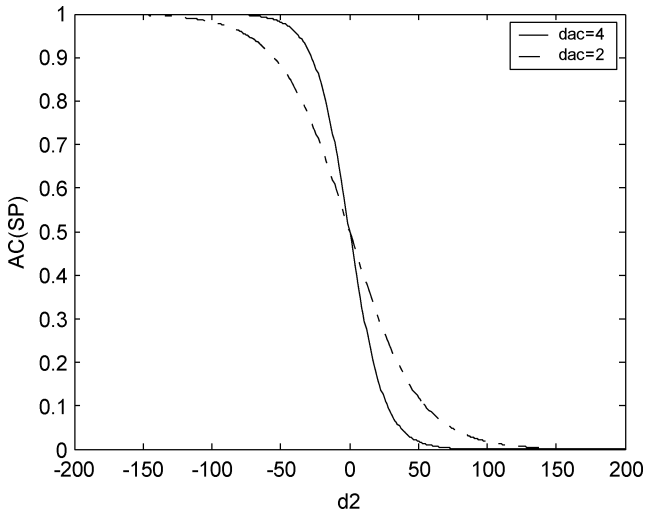

Fig. 10. Amplitude control penalty function $(dfr = 50\%)$.

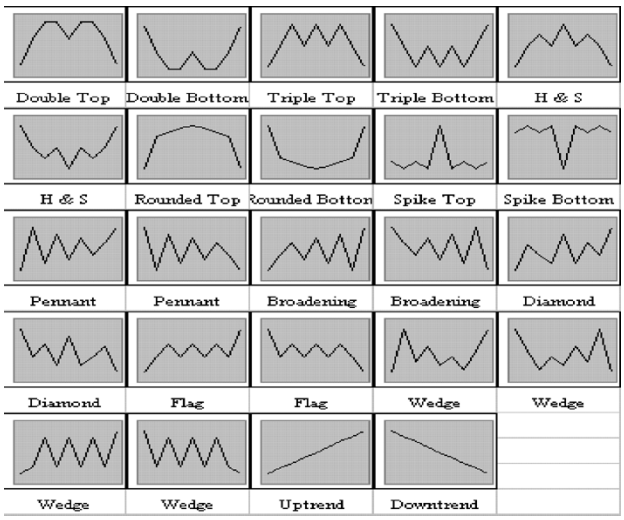

Fig. 11. Twenty-two technical patterns plus uptrend and downtrend.

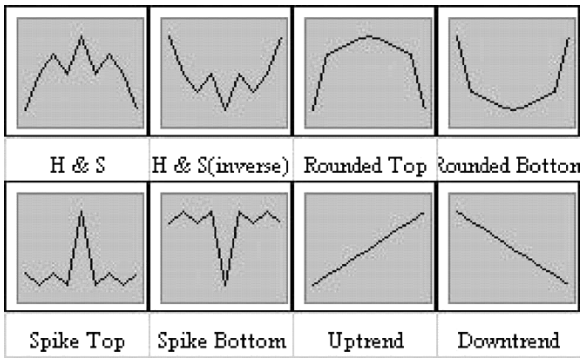

Fig. 12. Patterns for artificial time series generation.

TABLE I
PARAMETER VALUE ADOPTED

Parameter		Value
PopSize	Size of population	50
MaxGen	Maximum number of generation	1000
MinFitness	Minimum fitness value	0.0
R_c	Crossover rate	0.4
R_a	Probability of adding a time point during mutation	0.5
dlen	Desired length of subsequence	180

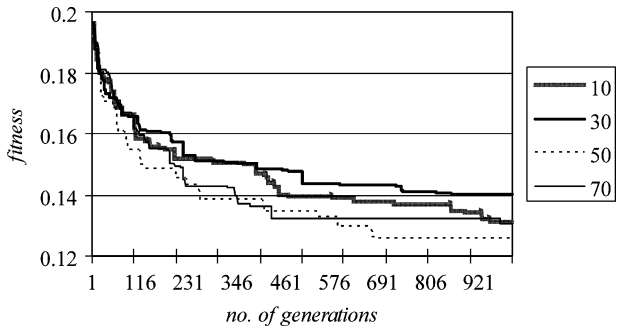

Fig. 13. Learning time of different population sizes.

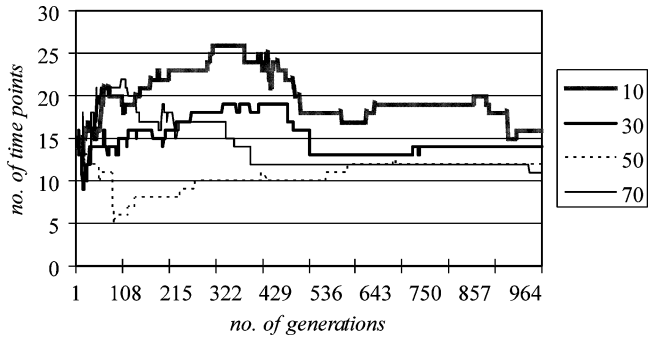

Fig. 14. Effects of different *PopSize* on the number of time points obtained.

TABLE II
FITNESS VALUES WITH DIFFERENT CROSSOVER RATES ($ds = 180$)

Crossover Rate R_c	No. of time points	Fitness value
0	14	0.139706
0.2	11	0.137088
0.4	12	0.125674
0.6	14	0.126418
0.8	33	0.136121
1	12	0.180087

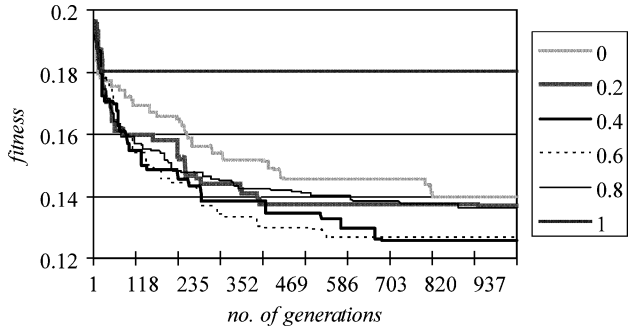

Fig. 15. Learning curves for different crossover rates.

For example, if the desired segment length $dlen$ is 180 (e.g., 180 trading days) and dlc is set to 2, then $\theta_1 = 90$, then the temporal control penalty function will look like Fig. 9. However, if dlc is set to 6 ($\theta_1 = 30$), the temporal control is strengthened (see Fig. 9) by adding a greater penalty to the fitness evaluations for patterns with a length different from the desired one.

B. Amplitude Control

On the other hand, in the amplitude domain, users may also want to control the desired minimum scale (i.e., fluctuation) of the patterns that they are focusing on. To achieve this goal, it is suggested that users can specify their preferred fluctuation rate (fr) for the patterns. Here, the fluctuation rate of a segment is defined as the percentage of the amplitude scale of the pattern compared with the minimum amplitude scale, that is

$$fr = \frac{\max_k(SP_k) - \min_k(SP_k)}{\min_k(SP_k)}. \tag{8}$$

The amplitude control penalty function is then defined as follows:

$$AC(SP) = 1 - \frac{1}{\left(1 + \exp^{-\frac{d_2}{\theta_2}}\right)} \tag{9}$$

where $d_2 = fr - dfr$, that is, the difference between the fluctuation rate (fr) of that segment and the desired minimum fluctuation rate (dfr), or the desired minimum amplitude scale specified by the users. Again, θ_2 is a parameter used to control the sharpness of the function, and it is defined as follows:

$$\theta_2 = \frac{dfr}{dac} \tag{10}$$

where dac is the desired amplitude control parameter. As in temporal control, larger values of dac will lead to smaller values of θ_2, which will then make the slope of the transition curve steeper. As illustrated in Fig. 10, for $dfr = 50\%$, the transition curve with $dac = 4$ is steeper than that with $dac = 2$.

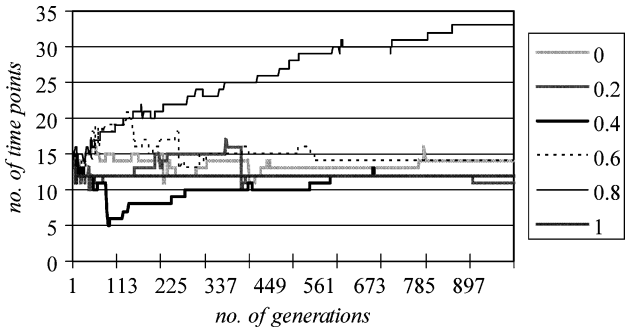

Fig. 16. Effects of different crossover rates on the number of time points obtained.

TABLE III
FITNESS VALUES WITH DIFFERENT $R_a(ds = 180)$

Probabilityof adding a time point R_a	No. of time points	Fitness value
0.1	8	0.147467
0.3	8	0.136291
0.5	12	0.125674
0.7	23	0.140939
0.9	43	0.139566

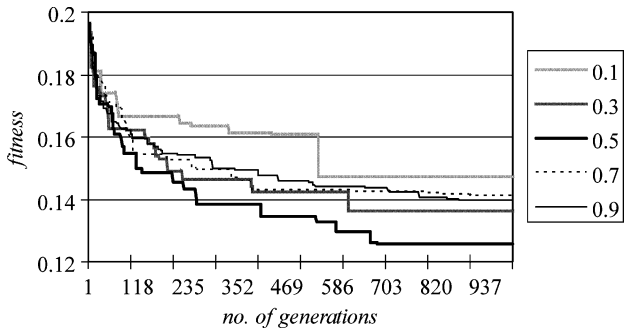

Fig. 17. Learning curves for different R_a.

With the introduction of the temporal and amplitude controls, the fitness function is modified as follows:

$$\text{fitness}(C_{i,j}) = \min_{\forall Q} \{DM(SP, Q) + TC(C_{i,j}) + AC(C_{i,j})\}. \quad (11)$$

C. Weighting Different Pattern Templates

With the distance measure defined in (4), we found that the pattern templates (e.g., the "flag" technical pattern) having more data points around the middle amplitude level, i.e., 0.5 in the normalized unit range, were favored in the fitness evaluation. This is because the expected error of those data points, arising from (2), is smaller. The expected error of data point Q_k can be calculated by

$$E_{\text{Err}}(Q_k) = \Pr(SP_k > Q_k) \times E_{\text{Err}}(Q_k | SP_k > Q_k) + \Pr(SP_k < Q_k) \times E_{\text{Err}}(Q_k | SP_k < Q_k) \quad (12)$$

where $\Pr(SP_k > Q_k)$ denotes the probability of $SP_k > Q_k$ and $E_{\text{Err}}(Q_k | SP_k > Q_k)$ is the expected error of data point Q_k given $SP_k > Q_k$. If we assume that SP_k is evenly distributed in the unit range, $E_{\text{Err}}(Q_k | SP_k > Q_k)$ will be equal to $0.5 \times (1 - Q_k)$, while $E_{\text{Err}}(Q_k | SP_k < Q_k)$ will be equal to $0.5 \times Q_k$. By estimating the expected error of each data point in a pattern template Q, the expected error for this pattern template can be computed by

$$E_{\text{Err}}(Q) = \frac{1}{n} \sum_{k=1}^{n} E_{\text{Err}}(Q_k). \quad (13)$$

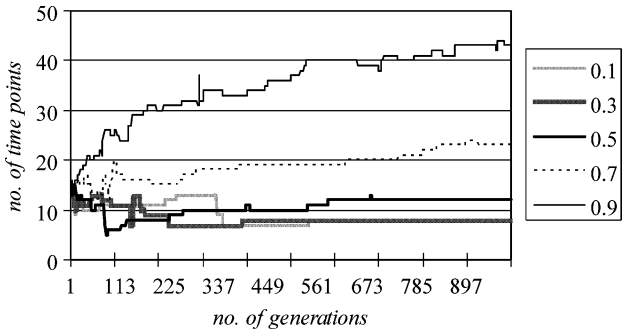

Fig. 18. Effects of different R_a on the number of time points obtained.

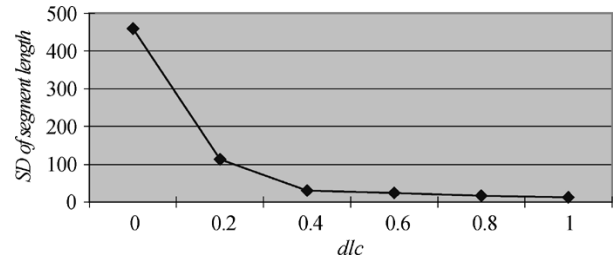

Fig. 19. Standard deviation of segment length for different dlc values.

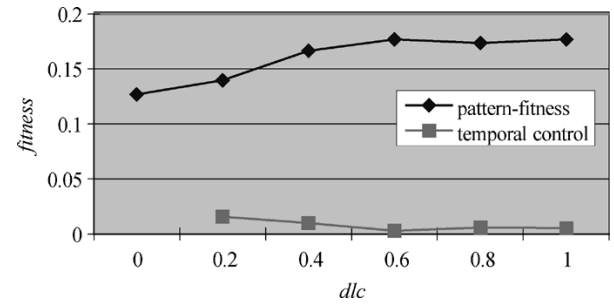

Fig. 20. Fitness values for different dlc values.

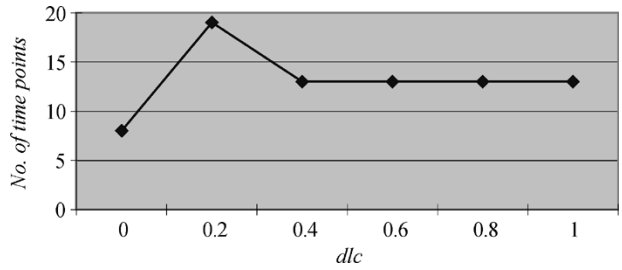

Fig. 21. Number of time points obtained for different dlc values.

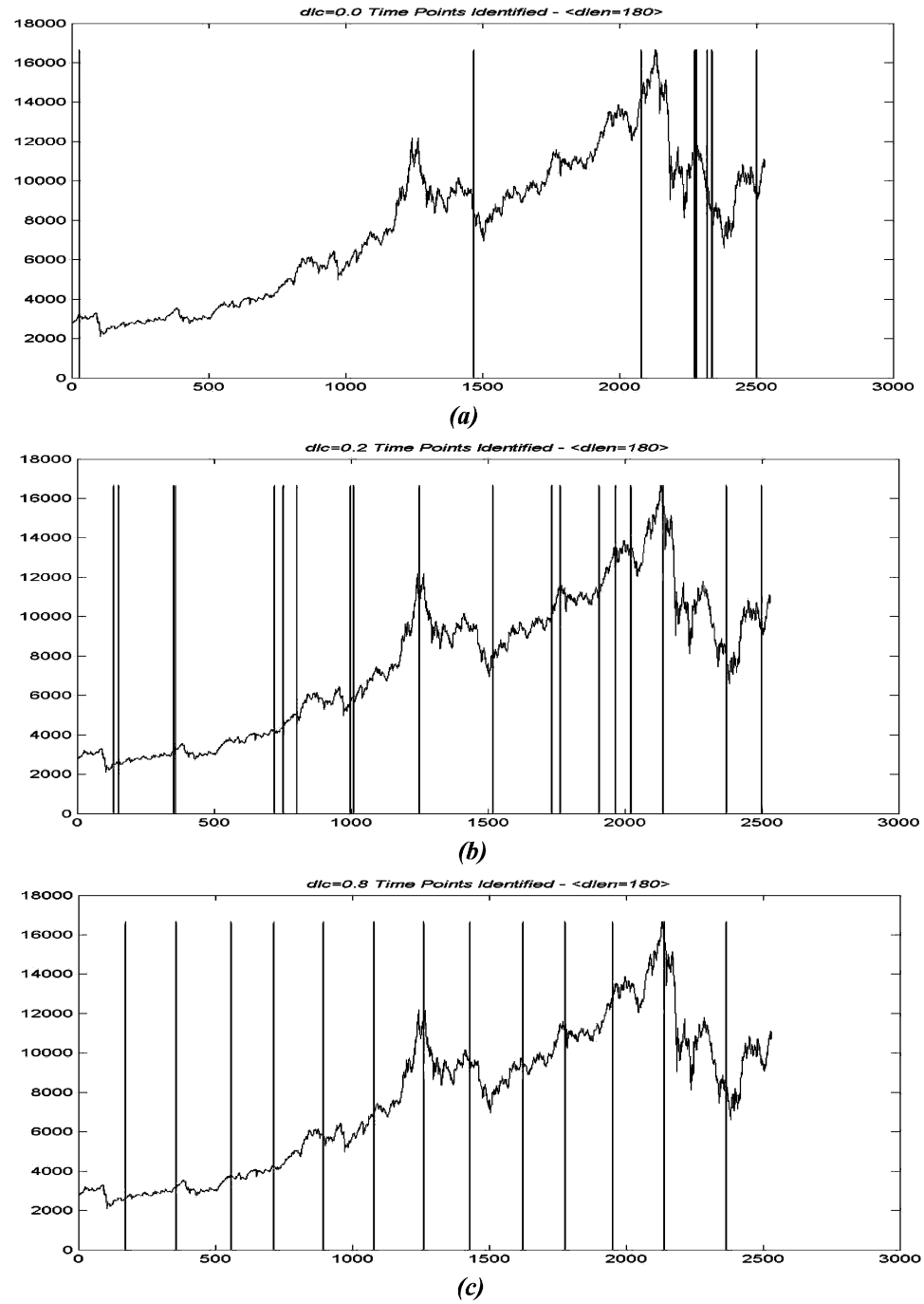

Fig. 22. Segmentation results for different dlc values. (a) $dlc = 0$. (b) $dlc = 0.2$. (c) $dlc = 0.8$.

To take the expected errors of different pattern templates into considerations, we introduce a template weighting term and formulate it as

$$TW(Q) = 1 + \left(1 - \frac{E_{\mathrm{Err}}(Q) - \min\limits_{\forall Q}\{E_{\mathrm{Err}}(Q)\}}{\max\limits_{\forall Q}\{E_{\mathrm{Err}}(Q)\} - \min\limits_{\forall Q}\{E_{\mathrm{Err}}(Q)\}}\right) \tag{14}$$

with values in the range of 1 to 2. Thus, the distance measure in (4) is redefined as

$$DM(SP,Q) = w_1 \times AD(SP,Q) \times TW(Q) + (1 - w_1) \times TD(SP,Q) \tag{15}$$

to make it unbiased toward different pattern templates. In addition, $TW(Q)$ can be further adjusted by users to fulfill their need besides following the guideline of calculating such weight in this section. For example, users may prefer some kind of patterns, in this case, they can decrease their weightings to increase the probability of their appearing in the final segmentation result.

V. Simulation Results

In this section, we report the performance of the proposed evolutionary time series segmentation algorithm. Two sets of experiments were conducted for parameter analysis. The first set

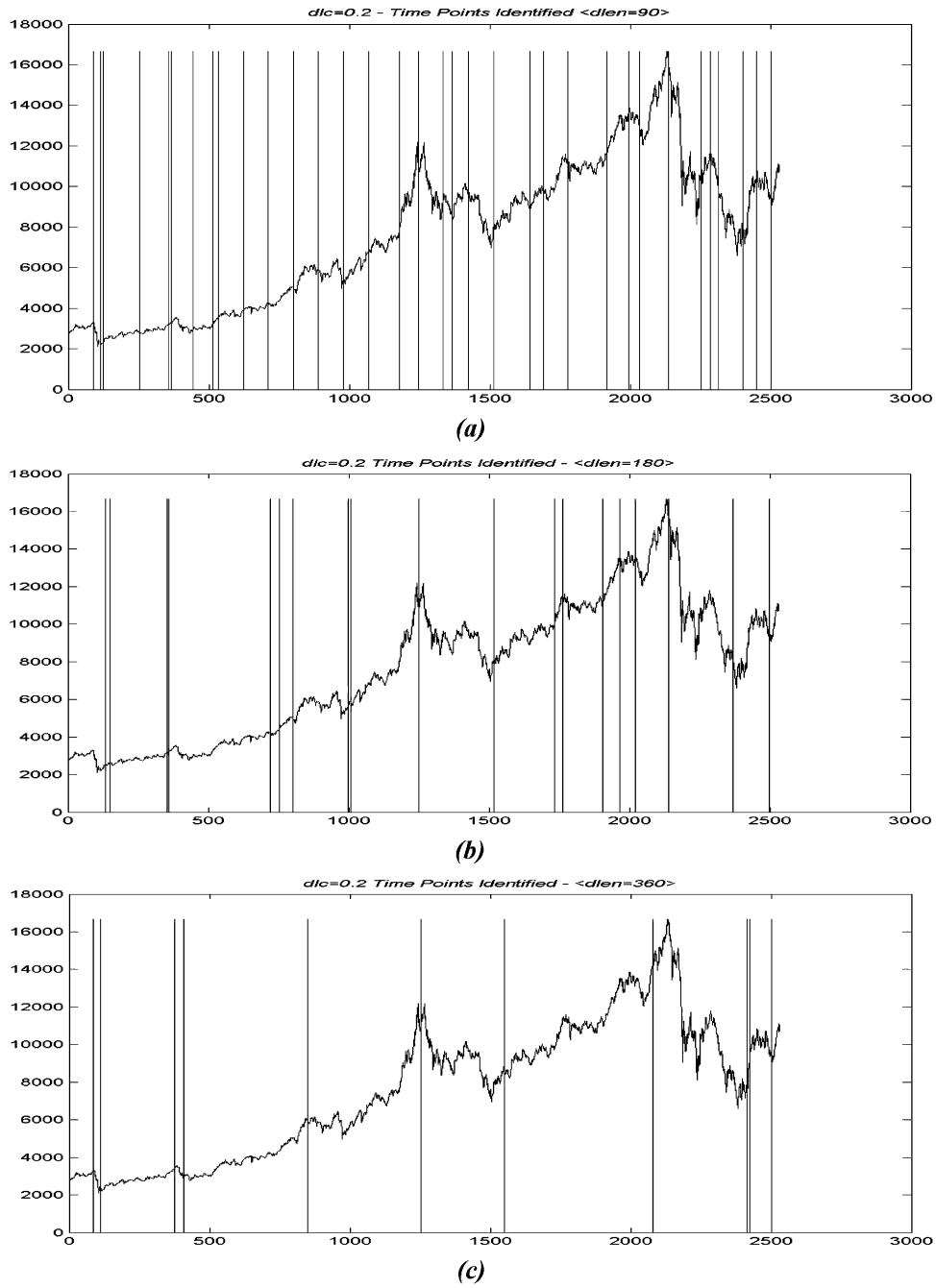

Fig. 23. Segmentation results of the Hong Kong Hang Seng Index time series. (a) $dlen = 90$. (b) $dlen = 180$. (c) $dlen = 360$.

was used to test the parameters of the evolutionary algorithm, whereas the second set was designed to validate the effects of the controllable parameters during the segmentation process. The results are reported in Sections V-A and V-B, respectively. The time series data used was taken from the daily opening price of the Hong Kong Hang Seng Index from March 1989 to March 1999 (i.e., 2532 data points). Twenty-two technical patterns were used as the pattern templates $\{Q\}$. Fig. 11 shows that their length is 9 (i.e., 9 points). As mentioned in Section II-A, these pattern templates have to be normalized to a unit interval for fitness computation.

In addition, two kinds of time series data were used to evaluate the segmentation results: artificial time series generated from the combinations of pattern templates and the time series of selected Hong Kong stocks. The first kind of time series data was used to demonstrate the segmentation effects using time series generated artificially with 500 data points, by combining patterns in Fig. 12; both their chance of appearing and pattern length are varying randomly. Moreover, noise is added to the generated time series.

The second kind of time series data used was taken from several opening price time series from different sectors of the Hong Kong Stock Market. However, due to limitations of paper length, only three are reported here. Unless otherwise specified, the parameters in this part are set to the default values shown in Table I. We also use these three stock time series to compare

the performance of our algorithm with another segmentation algorithm [20] and the results are reported in Section V-C. In the final subsection, a usage of the segmentation results in decision making is described.

A. Analysis of the Evolutionary Segmentation Process

1) Size of the Population: The size of the population (*PopSize*) is critical in the evolutionary process. If *PopSize* is large, more sets of time points will be introduced and evaluated, thus better fitness results can be obtained. This was verified, as shown in Fig. 13. In addition, larger *PopSize* causes the GA to converge faster than a smaller one. However, there is no further improvement after *PopSize* is larger than 50. As is well known, an increase in population size will require more computational resources. Fig. 14 also shows that the number of time points obtained is quite independent of the population size.

2) Crossover Rate: Table II shows the fitness values when different crossover rates are used. Figs. 15 and 16 show the learning curves and the effects on the number of time points obtained for different crossover rates. If the evolutionary process only depends on crossover (i.e., $\text{crossover rate} = 1.0$, hence, there is no mutation), no new genes (i.e., time points) will be introduced to the evolutionary process. Therefore, the process converges in the early stage with a bad fitness value. Better results can be obtained when the crossover rate is decreased and mutation is allowed. Fig. 16 shows that the number of time points obtained is fairly independent of the crossover rate. This is because the probability of adding a time point during mutation is actually the same probability as dropping a point (i.e., $R_a = 0.5$).

3) Probability of Adding a Time Point During Mutation: Table III shows the fitness values when different R_a, i.e., probabilities of adding a time point during mutation, are used. Figs. 17 and 18 show the learning curves and the effects on the number of time points obtained for different R_a. Larger number of time points are added with increase in R_a and *vice versa*. The best result obtained here indicates that a fair chance of adding or dropping a time point (i.e., $R_a = 0.5$) should be used.

B. Analysis of the Controllability

1) Temporal Control: With the desired segment length $dlen = 180$, Fig. 19 shows the standard deviation of the segment length when using different desired length control values dlc. As expected, the standard deviation will become small when dlc is large, (i.e., a sharp temporal control penalty function is used). Fig. 20 shows that increasing dlc leads to deterioration in fitness, while keeping temporal control penalty stable. Fig. 21 shows that the number of time points obtained is relatively insensitive to different values of dlc. If dlc increases, the number of time points tends to become the number that can segment the time series evenly (in this case, 2532 data points are divided by the desired segment length 180, i.e., 14 time points). From the segmentation results shown in Fig. 22, increasing dlc will lead to a segment length that is much closer to the $dlen$ specified by the users. When dlc is increased beyond 0.2, the time points are close to being evenly distributed across the time series, as shown in Fig. 22(c).

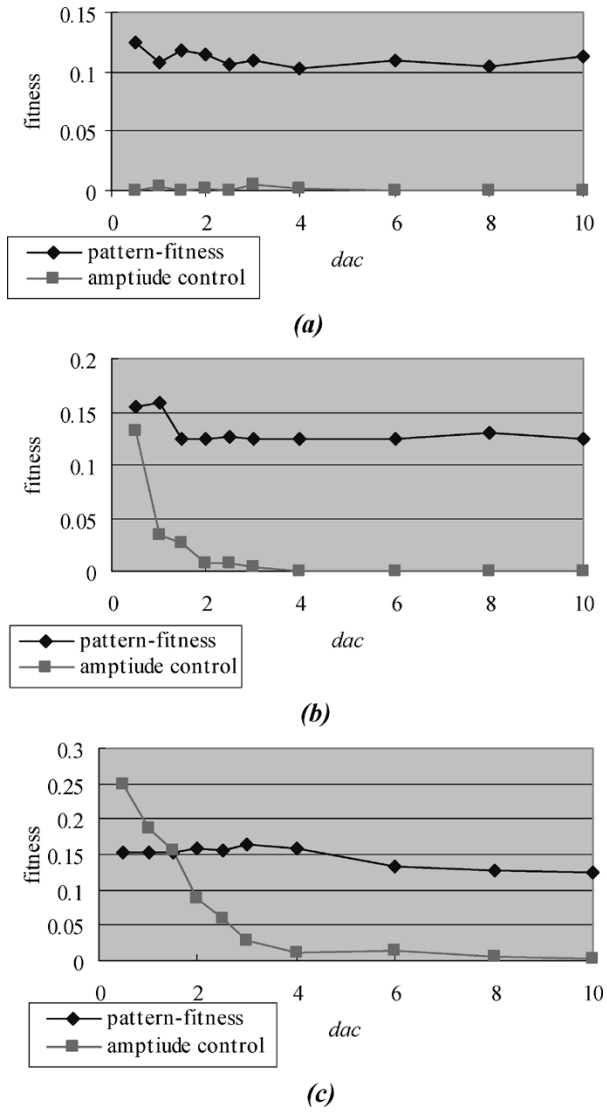

Fig. 24. Fitness for different dac values. (a) $dfr = 10$. (b) $dfr = 50$. (c) $dfr = 100$.

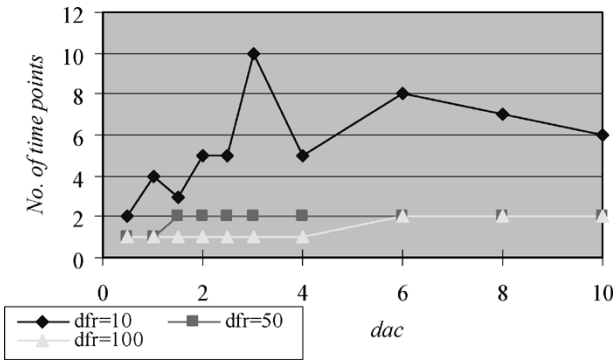

Fig. 25. Number of time points obtained for different dac values.

By fixing $dlc = 0.2$ and setting $dlen = 90$, 180 and 360, the average segment lengths are 73, 127, and 212, respectively. Fig. 23 shows their corresponding segmentation results.

2) Amplitude Control: Fig. 24 shows the effects on pattern fitness and amplitude control penalty of increasing the desired amplitude control dac. With a larger dfr, the amplitude control

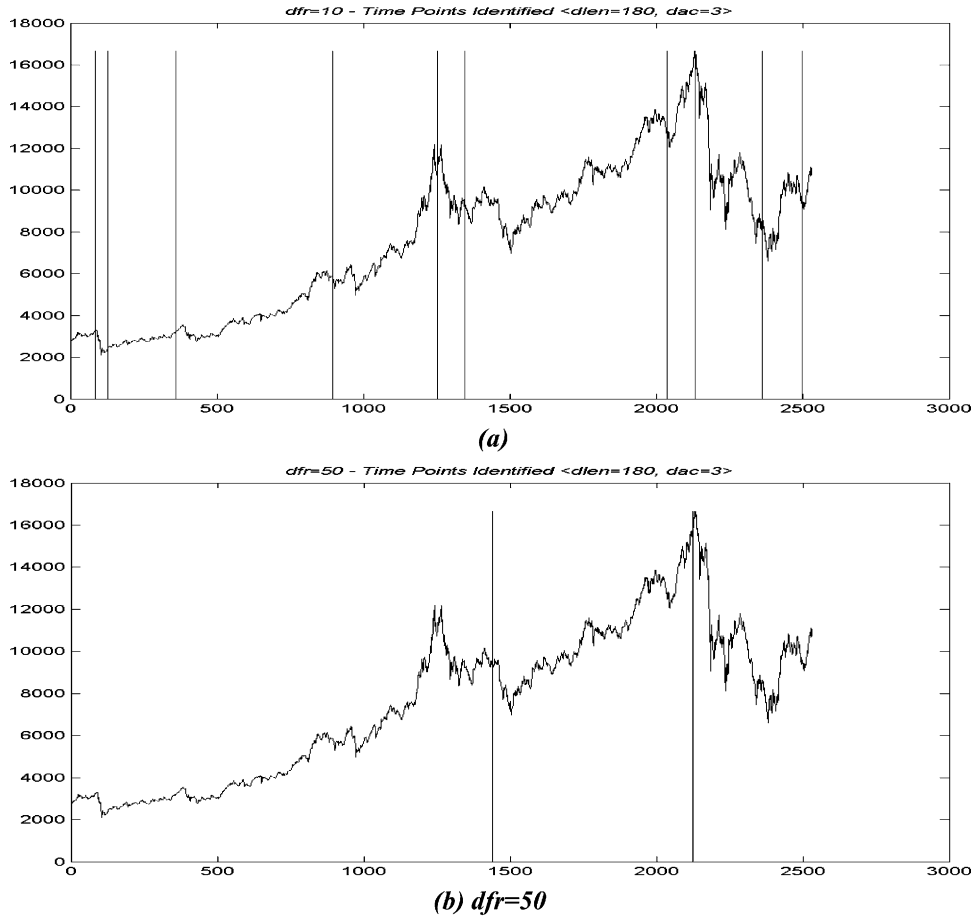

Fig. 26. Segmentation results of the Hong Kong Hang Seng Index time series. (a) $dfr = 10$. (b) $dfr = 50$.

penalty is larger due to the fact that it is more difficult to obtain the required fluctuation rate. We can clearly see this effect when $dfr = 100$ [Fig. 24(c)]. In this case, the amplitude control penalty is even higher than the pattern fitness for smaller dac values. As dac increases, the threshold effect leads to the segments having a smaller amplitude control penalty.

To achieve the goal of having segments with a larger amplitude scale, the evolutionary process tends to merge segments together to form a larger segment with larger fluctuation rate. This is the reason why the number of time points decreases with a increasing dfr, as shown in Fig. 25. Furthermore, when the number of time points cannot be further decreased (due to pattern fitness), the amplitude control penalty increases rapidly with increasing dfr (see Fig. 24). Fig. 26 shows the final segmentation results using $dfr = 10$ and 50. Larger segments with larger amplitude differences can be found when $dfr = 50$.

3) Weighting of Pattern Templates: As discussed in Section IV-C, with the distance measure defined, pattern templates having more data points around the middle amplitude level were favored in the fitness evaluation. Uptrend and downtrend are two of the preferred templates under this distance measure. Furthermore, they are the primitive shape for the construction of other pattern templates. As shown in Fig. 27(a), the appearance of these two patterns always dominated in the final segmentation (N/S conversion) results. To solve this problem, template weighting is applied during distance measure to make it unbiased toward different pattern templates. Fig. 27(b) shows the segmentation result when template weighting is applied. The circled subsequences are identified as "spike bottom" and "diamond," respectively, and all the rest are identified as either uptrend or downtrend as in Fig. 27(a). In addition, template weighting can be further adjusted by users to fulfill their requirements. For example, Fig. 28 shows the results when the weightings of both "head-and-shoulder" and "inverse head-and-shoulder" patterns are decreased to increase their probability of appearing in the final segmentation result. As expected, two "inverse head-and-shoulder" patterns are identified.

C. Further Analysis

1) Segmentation of Artificial Time Series: In this experiment, the performance of segmenting artificial time series is demonstrated. Figs. 29–31 show the three artificial time series (consisting of 500 data points) generated by random combination of resized pattern templates in Fig. 12 using uniform time scaling as depicted in Fig. 32. Figs. 29(a), 30(a), and 31(a) mainly focus on the abilities of the proposed algorithm in pattern identification. So, only six reversal technical patterns are used (see Fig. 12 and ignore the uptrend and downtrend templates). The circled segments indicate that the error rate (fitness) is less than 0.05. We can see that most of the reversal technical patterns can be identified even in different resolutions.

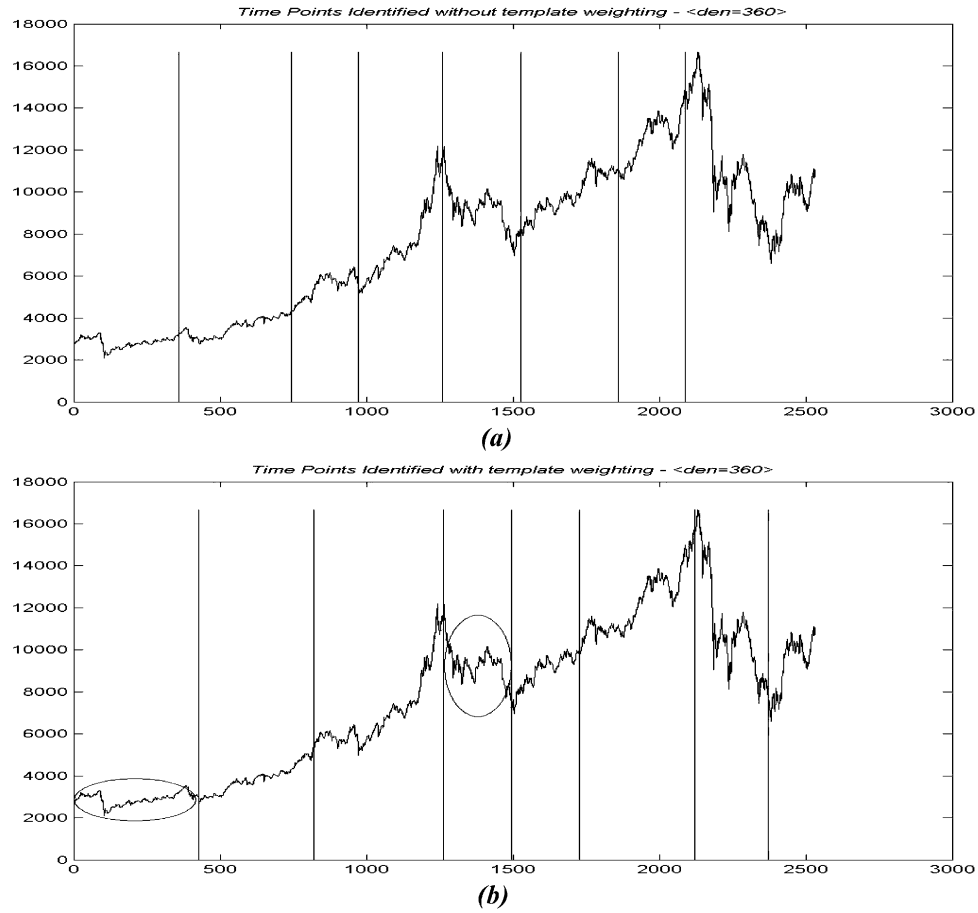

Fig. 27. Segmentation results of the Hong Kong Hang Seng Index time series (a) without and (b) with template weighting of pattern templates.

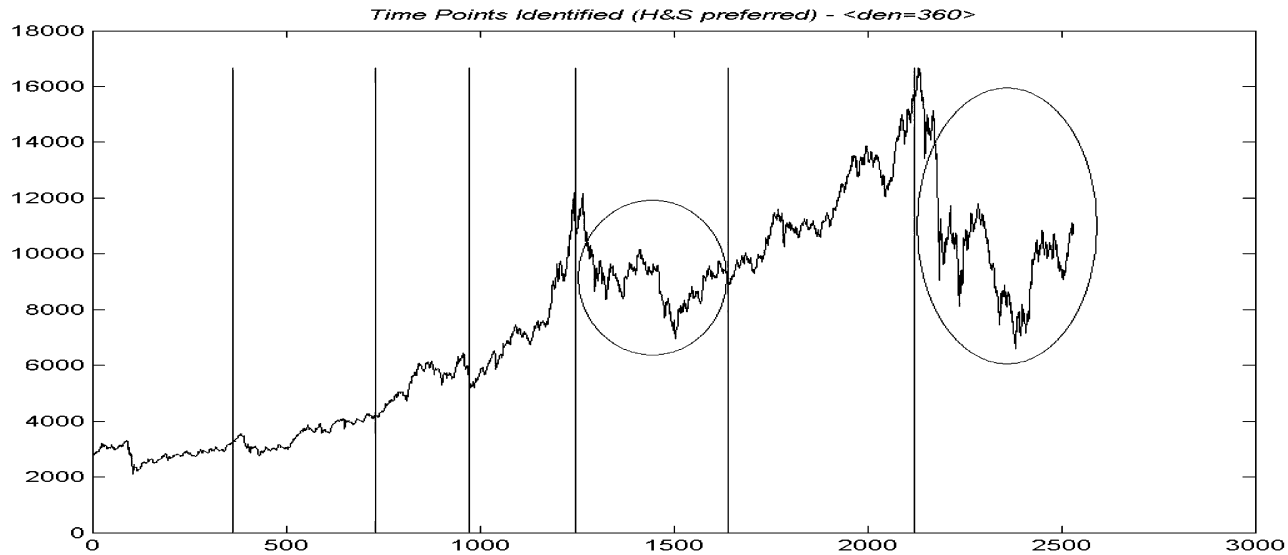

Fig. 28. Segmentation results of the Hong Kong Hang Seng Index time series with decreased weighting of "head-and-shoulder" and "inverse head-and-shoulder" patterns.

Figs. 29(b), 30(b), and 31(b) are the segmentation results of the artificial time series with noise added when only six pattern templates are used. Adding noise is controlled by two parameters, namely, the probability of adding noise for each data point (α) and the level of noise being added to such point (β), as shown in Fig. 32. Although the results are not as good as that of the clean series, most of the technical patterns involved can still be identified.

2) Segmentation of Real-Stock Time Series: In this section, we report the investigation of three more closing price time series from the Hong Kong stock market. All the parameters are set to the default values shown in Table I if they are not

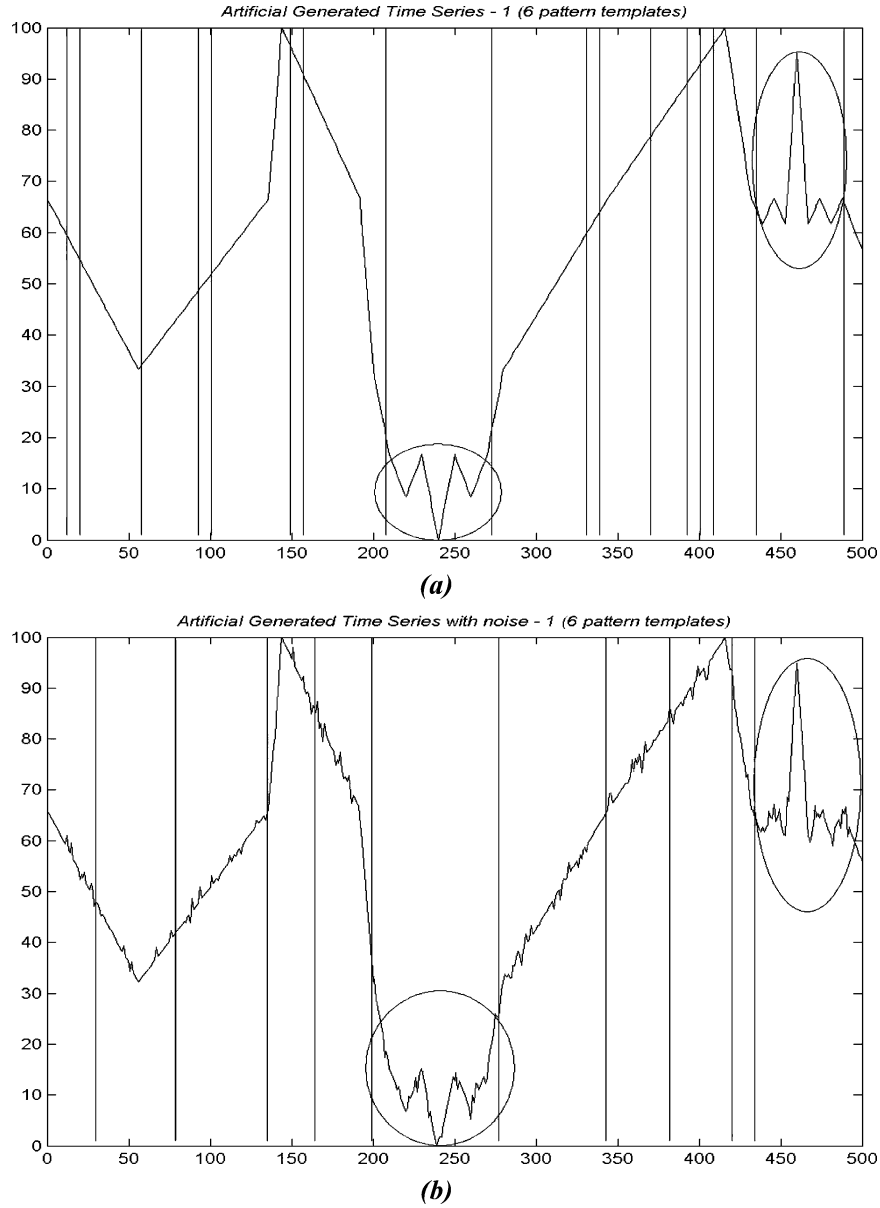

Fig. 29. Segmentation results of an artificial time series 1. (a) By the proposed algorithm using six pattern templates (without noise). (b) By the proposed algorithm using six pattern templates (with noise added).

specified. From the estimation in the experiments, dlc is set to 0.3. The same 22 technical patterns (Fig. 11) are used as the pattern templates $\{Q\}$ in results Figs. 33(a) and (b)–35(a) and (b). By setting different $dlen$ values (i.e., 180 and 360), different segmentation results with different considerations are shown. With larger $dlen$, long-term stock price movement is considered because larger segments are produced with a smaller number of time points [Figs. 33(b)–35(b)]. We observe that the time points identified using different $dlen$ are independent [comparing the results of Figs. 33–35(a) with Figs. 33–35(b)]. The time points are identified for the best matching of the segments with pattern templates. As in the previous subsection, the same set of experiments ran again with only six reversal technical patterns provided for pattern identification. The circled segments indicate that the error rate (fitness) is less than 0.05. Again, obvious reversal technical patterns can be identified in different resolutions.

3) Performance Comparison: In this section, the performance of the proposed evolutionary segmentation algorithm is compared with that of an existing technique. We acknowledge that the comparison here is not very systematic because there exists a fundamental difference between our algorithm and the traditional ones. That is, the proposed method is a template driven segmentation approach, which requires a given set of pattern templates, while most existing segmentation methods are dealing with the time series itself, such as detecting special events in the time series as the cutting points [7], minimum message length segmentation [5], and segmentation by piecewise linear approximation (PLA) [20]. The segmentation results of [20] are shown in Figs. 36 and 37. The time series are segmented by approximating the time series with straight lines and merging the lowest cost segments until the required number of segments is produced. That means the number of segments preferred must be specified beforehand. It is related

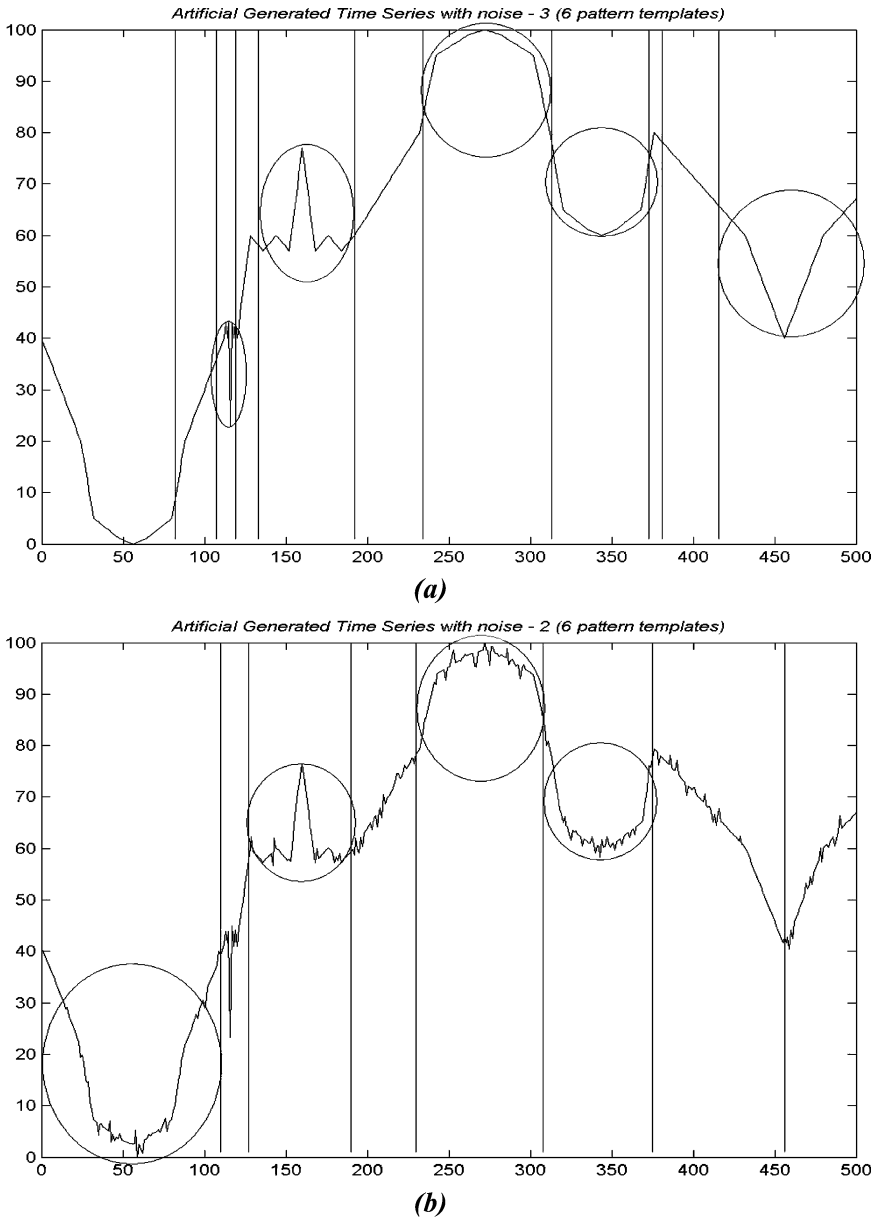

Fig. 30. Segmentation results of an artificial time series 2. (a) Proposed algorithm using six pattern templates (without noise). (b) Proposed algorithm using six pattern templates (with noise added).

to our desired length $dlen$ and, hence, there are nine and six segments formed with respect to $dlen = 180$. As can be seen from Figs. 36 and 37, without the knowledge of pattern templates for segmentation leads to the missing of special patterns (e.g., head-and-shoulder pattern) that stock analysts are looking for. It is because the stock time series is segmented by minimizing the piecewise linear approximation error rate. The segmentation results have been biased toward generating uptrend and downtrend. In fact, most existing segmentation algorithms will behave in a similar way and, hence, are not suitable for financial time series segmentation. For the proposed evolutionary segmentation algorithm, the segmentation results have already been depicted in Fig. 33(c) and Fig. 34(c), where interesting technical patterns can be identified. We believe that such a segmentation result is more preferable to the end users.

D. On the Use of Segmentation Results

The time series segmentation results are typically used to index and query (search) the time series database [21]. As to our focus on stock data analysis, this section discusses a usage of the segmentation results in investment decisions. Recall from [22] that technical analysis is concerned with what has actually happened in the market rather than what would happen. The technical analyst is not concerned with the "bigger picture" factors affecting the market, as is the fundamental analyst, but concentrates on the activity of that instrument's market. Charting the prices of a financial instrument creates a picture of the market "battle" between buyers (bulls) and sellers (bears). In order to decide if they should buy or sell, market players need to investigate the occurrence of technical patterns so that the market trends, e.g., uptrend, downtrend, and continuation, can be determined or predicted. Uptrend and downtrend are intu-

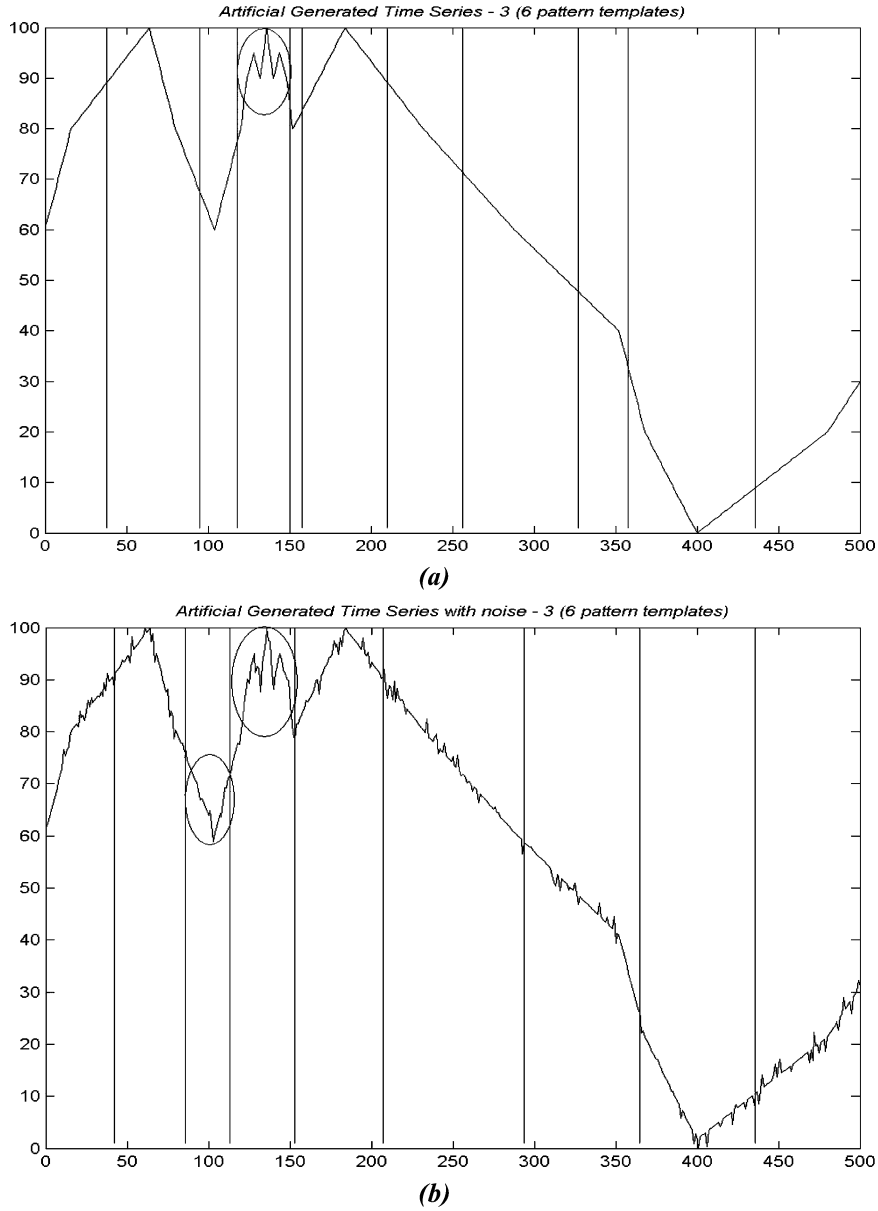

Fig. 31. Segmentation results of an artificial time series 3. (a) Proposed algorithm using six pattern templates (without noise). (b) Proposed algorithm using six pattern templates (with noise added).

Uniform Time Scaling

Uniform time scaling of the pattern templates from $P[1..n]$ to $P[1..m]$, where $n=9$ and $m=n\times r$ $(1\le r\le 10)$

Noise Adding

```
For each data point P[i] in P
   If (randomly generated probability <α)
      diff = P[i]×(random_value between 0 to β);
      P[i] = P[i] ± diff;
   End If
End For
```

Fig. 32. Procedures used for artificial time series generation.

itive. Continuation occurs during periods of consolidation when prices are moving sideways following an uptrend or downtrend. The patterns are not always easy to recognize and do not always have regular shapes. Continuation patterns have names based on geometric shapes, e.g., broadening, wedge, flag, and pennant. They last for varying periods of time—flags and pennants only last a few days, while wedge can last up to a year. Market players use continuation patterns to determine a target price for their trading strategy. This target price is the level they expect the market to reach following a breakout of the consolidation and a resumption of the continuation trend.

Based on the segmentation results, we investigate the chance of a particular movement being developed after a specified pattern appeared. The 22 commonly used technical patterns are classified into three categories, which indicate the "immediate" movement of the market price: Up—double top, triple top, H&S, rounded top, flag (upward), and uptrend; Down—double bottom, triple bottom, inverse H&S, rounded bottom, flag (downward), and downtrend; and Continuation—spike top

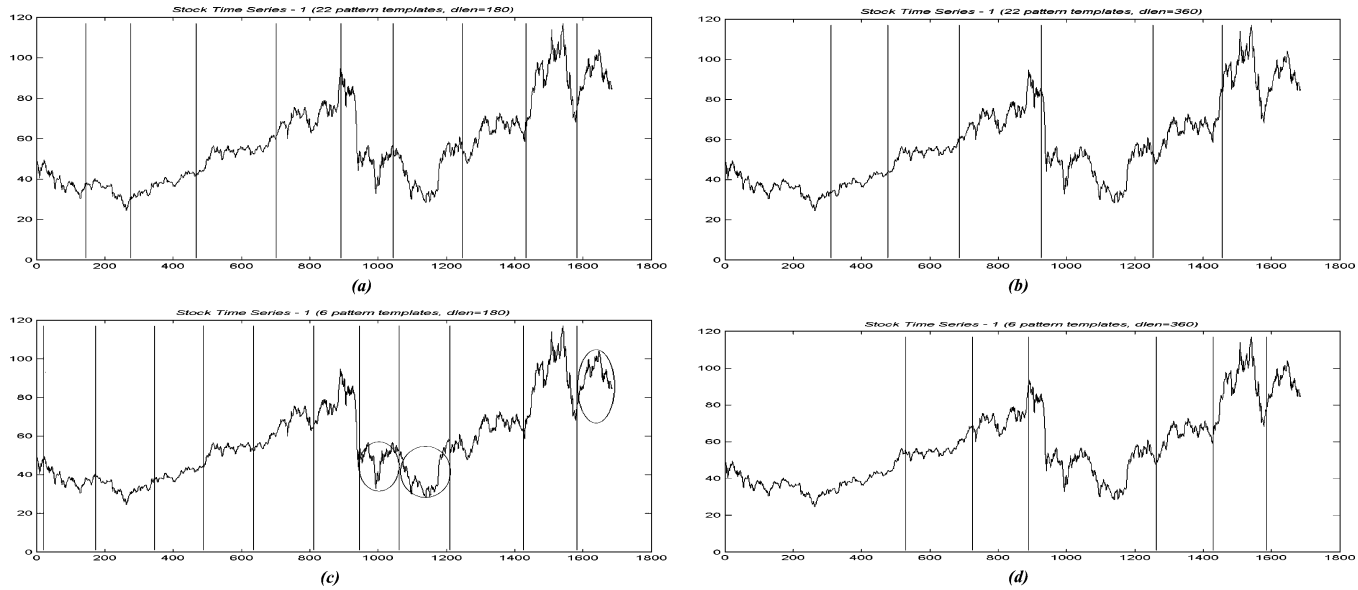

Fig. 33. Segmentation results of stock time series 0001 for different $dlen$ and different number of templates. (a) $dlen = 180$ (22 pattern templates). (b) $dlen = 360$ (22 pattern templates). (c) $dlen = 180$ (6 pattern templates). (d) $dlen = 360$ (6 pattern templates).

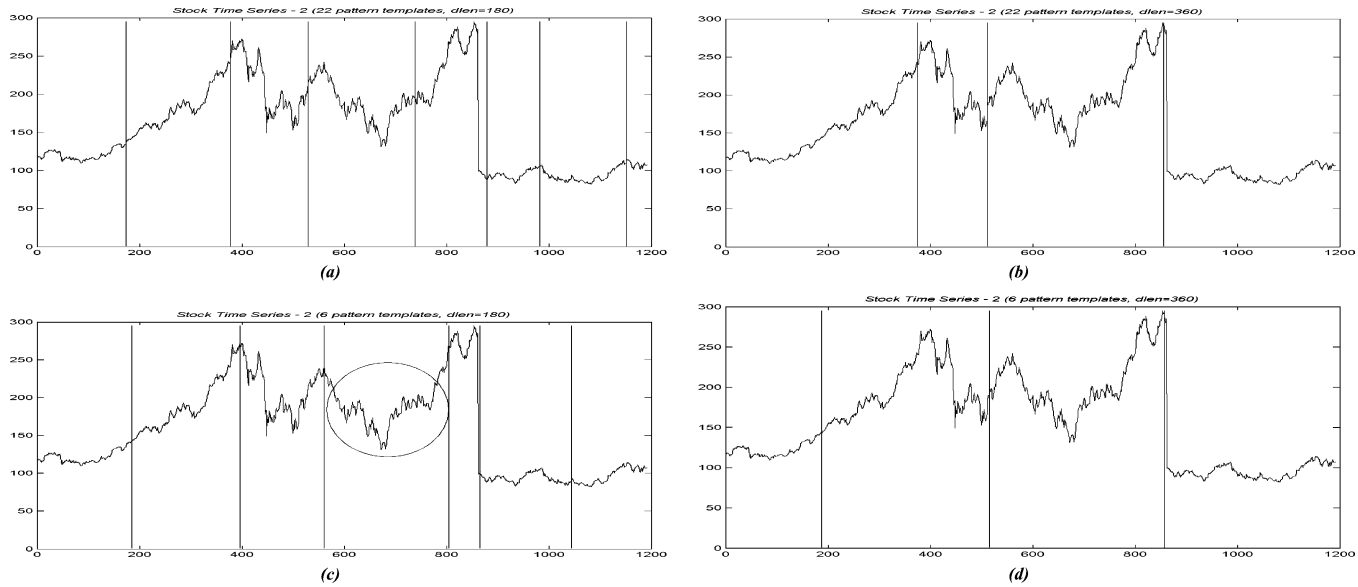

Fig. 34. Segmentation results of stock time series 0002 for different $dlen$ and different number of templates. (a) $dlen = 180$ (22 pattern templates). (b) $dlen = 360$ (22 pattern templates). (c) $dlen = 180$ (six pattern templates), (d) $dlen = 360$ (six pattern templates).

and bottom, pennant, broadening, and wedge. Taking double top as an example, the first one-third of the pattern shows an increase in stock price, i.e., an uptrend. On the other hand, the first half of rounded bottom indicates a downtrend, while the initial portion of spike top pattern can be considered as a continuation signal. In this experiment, we segment the three real-stock prices from 1994 to 2000 used in Section V-C2. Three resolutions corresponding to desired lengths 30, 90, and 180 were applied. Based on the sequence of patterns found after segmentation, the movement after a specified pattern appeared is investigated and the results have been summarized in Table IV. For the pattern just occurred, i.e., the first column of the table, we have limited ourselves to uptrend, downtrend, and the so-called "reversal patterns," including double top, triple top,. . ., inverse head-and-shoulder, and rounded bottom, because they should give us more information on the future price movement. As we can see, with a reversal pattern formed, the direction of movement has a higher chance in the opposite direction of the last movement of the pattern. For example, the occurrence of a rounded top pattern (downtrend on the latter part of the pattern) has led to an uptrend, which might be a double top, triple top, H&S, rounded top, flag (upward), or uptrend pattern here. With respect to our experiment here, such a relationship is always true, i.e., with 100% support from the rounded top patterns occurred. This gives the market players an important hint in making decisions when these kinds of patterns are formed. On the other hand, the price movements observed after an uptrend and a downtrend patterns are recorded in Table IV for references. It can be seen from the statistics that change of trend is more probable.

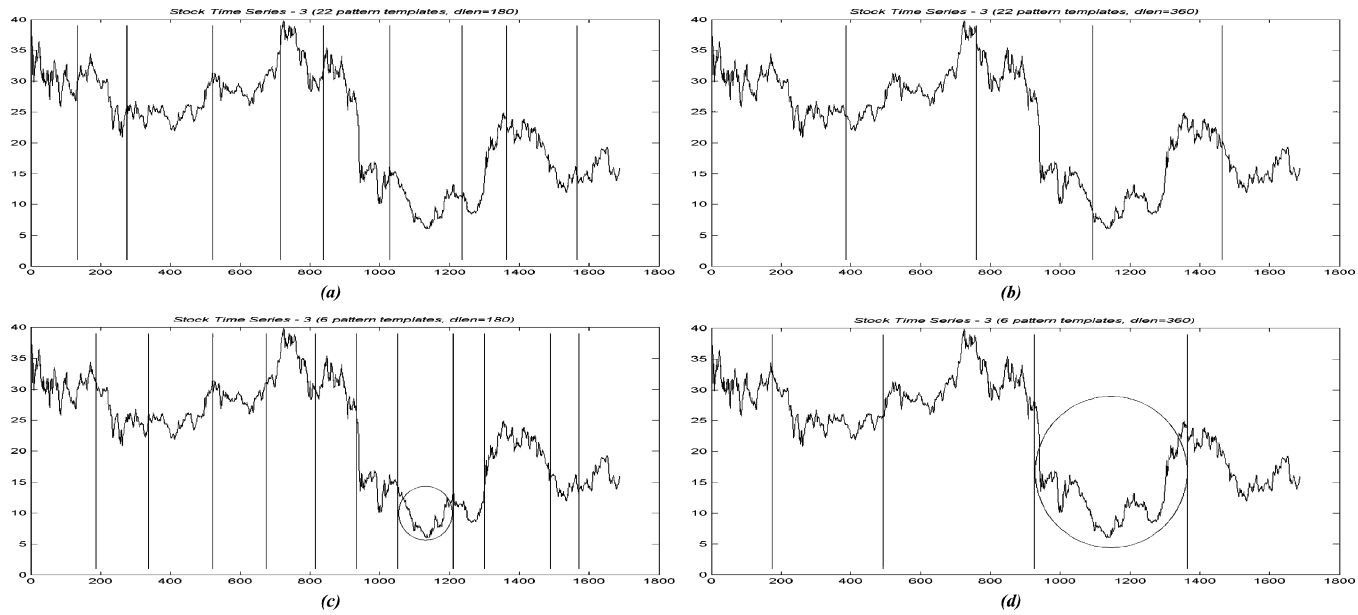

Fig. 35. Segmentation results of stock time series 0003 for different $dlen$ and different number of templates. (a) $dlen = 180$ (22 pattern templates). (b) $dlen = 360$ (22 pattern templates). (c) $dlen = 180$ (six pattern templates). (d) $dlen = 360$ (six pattern templates).

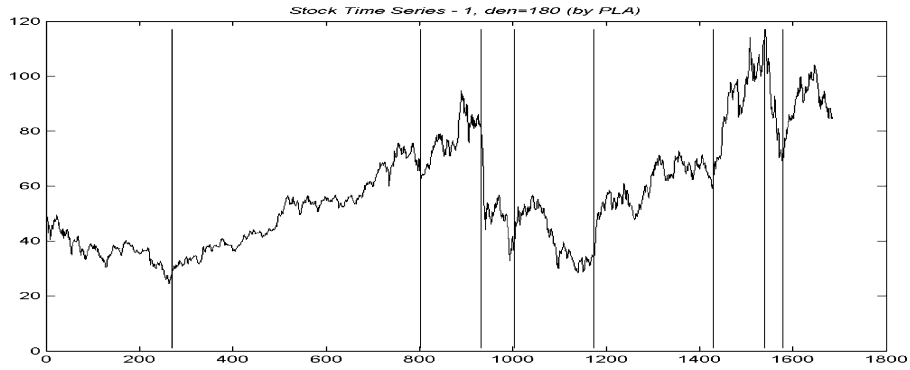

Fig. 36. Segmentation results of [20] for stock time series 0001.

TABLE IV
MOVEMENT OBSERVED FOR PATTERN JUST OCCURRED. THE PATTERNS AND MOVEMENT SIGNALS WERE EXTRACTED FROM THE EVOLUTIONARY SEGMENTATION RESULTS OF THREE REAL-STOCK PRICE TIME SERIES

Pattern Just Occurred	Movement Observed		
	Up	Down	Continuation
Double Top	60.0%	30.0%	10.0%
Triple Top	100.0%	0%	0%
H & S	75.0%	25.0%	0%
Rounded Top	100.0%	0%	0%
Double Bottom	14.3%	57.1%	28.6%
Triple Bottom	0%	100.0%	0%
Inverse H & S	23.5%	70.6%	5.9%
Rounded Bottom	0%	100.0%	0%
Uptrend	35.1%	56.4%	8.5%
Downtrend	54.6%	34.3%	11.1%

VI. CONCLUSION

In this paper, we propose an evolutionary segmentation algorithm for dividing a given time series into subsequences in accordance with a given set of pattern templates. Encouraging results are reported. Although the simulation results of this paper focus mainly on stock time series, the proposed algorithm can also be applied to other domains with a given set of pattern templates. In addition, a flexible time series pattern matching method based on perceptually important points is introduced for fitness evaluation. This method makes use of a time domain approach to carry out the matching process and it is intuitive for ordinary data analysts. One may find it particularly attractive in applications such as stock data analysis as shown here. The new method is efficient and effective for fitness evaluation in the evolutionary segmentation algorithm. Moreover, parameters are introduced to control both the temporal and amplitude scales of the final segments obtained and also the probability of pattern appearance. This will help to create an informative pattern space for subsequent query or mining activities.

By investigating or analyzing the history, the behavior of the time series can be obtained. The proposed segmentation algorithm contributes to provide a pattern-based summarization of the time series, which is particularly useful for time series indexing and query applications. We have not yet focused our work on time series forecasting in this paper. Like predicting the

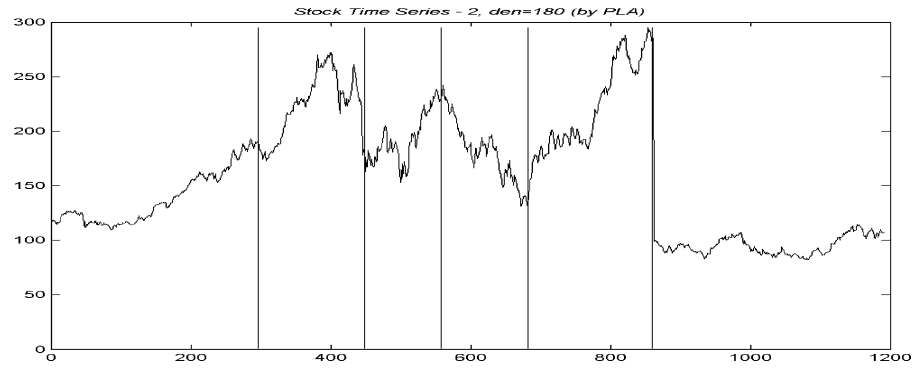

Fig. 37. Segmentation results of [20] for stock time series 0002.

future movement before a specific pattern is fully materialized, it is actually a task to analyze the formation of a pattern in the latest segment of the time series (last window) with new coming data points. This requires a partial pattern matching technique so that the possibility or probability of forming a specific pattern can be estimated. This will provide an early signal to the market players for making, say, investment decisions. Our future research work will move toward this direction.

REFERENCES

[1] H. Shatkay and S. B. Zdonik, "Approximate queries and representations for large data sequences," in *Proc. Int. Conf. Data Engineering*. Los Alamitos, CA: IEEE Computer Society Press, 1996, pp. 536–545.

[2] G. Das, K. I. Lin, and H. Mannila, "Rule discovery from time series," in *Proc. ACM SIGKDD Int. Conf. Knowledge Discovery Data Mining*, 1998, pp. 16–22.

[3] O. Y. Kai, W. Jia, P. Zhou, and X. Meng, "A new approach to transforming time series into symbolic sequences," in *Proc. 1st Joint BMES/EMBS Conf.*, Oct. 1999, vol. 2, p. 974.

[4] T. C. Fu, F. L. Chung, V. Ng, and R. Luk, "Evolutionary segmentation of financial time series into subsequences," in *Proc. Congr. Evolutionary Computation*, Seoul, Korea, 2001, pp. 426–430.

[5] J. J. Oliver, R. A. Baxter, and C. S. Wallace, "Minimum message length segmentation," in *Proc. Pacific-Asia Conf. Knowledge Discovery Data Mining*, 1998, pp. 222–233.

[6] J. J. Oliver and C. S. Forbes, "Bayesian approaches to segmenting a simple time series," Dept. Comput. Sci., Monash Univ., Victoria, Australia, Tech. Rep. 97/336, 1997.

[7] V. Guralnik and J. Srivastava, "Event detection from time series data," in *Proc. 5th ACM SIGKDD Int. Conf. Knowledge Discovery Data Mining*, 1999, pp. 33–42.

[8] A. N. Srivastava and A. Weigend, "Improving time series segmentation with gated experts through annealing," Dept. Comput. Sci., Inst. Cognitive Sci., Univ. Colorado, Boulder, CO, Tech. Rep. CU-CS-795-95, 1996.

[9] G. F. Bryant and S. R. Duncan, "A solution to the segmentation problem based on dynamic programming," in *Proc. 3rd IEEE Conf. Control Applications*, vol. 2, 1994, pp. 1391–1396.

[10] S. R. Duncan and G. F. Bryant, "A new algorithm for segmenting data from time series," in *Proc. 35th IEEE Conf. Decision Control*, vol. 3, 1996, pp. 3123–3128.

[11] A. N. Srivastava, R. Su, and A. S. Weigend, "Data mining for features using scale-sensitive gated experts," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 21, pp. 1268–1279, Dec. 1999.

[12] J. Han, W. Gong, and Y. Tin, "Mining segment-wise periodic patterns in time-related databases," in *Proc. Int. Conf. Knowledge Discovery Data Mining*, 1998, pp. 214–218.

[13] J. Han, G. Dong, and Y. Yin, "Efficient mining of partial periodic patterns in time series database," in *Proc. 15th Int. Conf. on Data Engineering*: IEEE Computer Society Press, 1999, pp. 106–115.

[14] C. L. Fancoua and J. C. Principe, "A neighborhood map of competing one step predictors for piecewise segmentation and identification of time series," in *Proc. IEEE Int. Conf. Neural Networks*, vol. 4, 1996, pp. 1906–1911.

[15] X. Ge and P. Smyth, "Deformable Markov model templates for time-series pattern matching," in *Proc. ACM SIGKDD Int. Conf. Knowledge Discovery Data Mining*, 2000, pp. 81–90.

[16] F. L. Chung, T. C. Fu, R. Luk, and V. Ng, "Flexible time series pattern matching based on perceptually important points," in *Proc. Int. Joint Conf. Artificial Intelligence Workshop (Learning From Temporal and Spatial Data)*, Seattle, WA, Aug., 4–10 2001, pp. 1–7.

[17] T. Back and H. P. Schwefel, "Evolutionary computation: an overview," in *Proc. IEEE Int. Conf. Evolutionary Computation*, 1996, pp. 20–29.

[18] T. Back, U. Hammel, and H. P. Schwefel, "Evolutionary computation: comments on the history and current state," *IEEE Trans. Evol. Comput.*, vol. 1, pp. 3–17, Apr. 1997.

[19] K. H. Liang, X. Yao, C. Newton, and D. Hoffman, "A new evolutionary approach to cutting stock problems with and without contiguity," *Comput. Oper. Res.*, vol. 29, no. 12, pp. 1641–1659, 2002.

[20] E. Keogh, S. Chu, D. Hart, and M. Pazzani, "An online algorithm for segmenting time series," in *Proc. IEEE Int. Conf. Data Mining*, 2001, pp. 289–296.

[21] S. Perng, H. Wang, S. R. Zhang, and D. S. Parker, "Landmarks: a new model for similarity-based pattern querying in time series databases," in *Proc. IEEE Int. Conf. Data Engineering*, 2000, pp. 33–42.

[22] The Reuters Financial Training Series, *An Introduction to Technical Analysis*. New York: Wiley, 1999.

Fu-Lai Chung (M'95) received the B.Sc. degree from the University of Manitoba, Winnipeg, MB, Canada, in 1987, and the M.Phil. and Ph.D. degrees from the Chinese University of Hong Kong, Shatin, in 1991 and 1995, respectively.

He joined the Department of Computing, Hong Kong Polytechnic University, Kowloon, in 1994, where he is currently an Associate Professor. He has published widely in the areas of soft computing, data mining, machine intelligence, and multimedia. His current research interests include time series data mining, bioinformatics data mining, fuzzy neural network modeling, and novel computational intelligence techniques.

Tak-Chung Fu received the B.A. degree in computing and the M.Phil. degree from the Department of Computing, Hong Kong Polytechnic University, Kowloon. He is currently working toward the Ph.D. degree in the Department of Computing, Hong Kong Polytechnic University.

His current research interests include time series analysis, data mining, and information retrieval.

Vincent Ng received the B.Sc. degree in mathematics and computing science from Simon Fraser University, Burnaby, BC, Canada, in 1982, the M.S. degree in mathematics from the University of Waterloo, Waterloo, ON, Canada, in 1986, and the Ph.D. degree from Simon Fraser University in 1994.

At present, he is an Associate Professor in the Department of Computing, Hong Kong Polytechnic University, Kowloon. His research interests include databases, data mining, and medical imaging.

Robert W. P. Luk (S'85–M'87–SM'01) received the B.Sc. degree in electronic engineering, the Engineering Diploma, the M.Sc. degree in cognition, computing, and psychology from the Department of Psychology, University of Warwick, Coventry, U.K., and the Ph.D. degree from the School of Electronics and Computer Science, University of Southampton, Southampton, U.K.

He is currently an Associate Professor in the Department of Computing, Hong Kong Polytechnic University, Kowloon. He was a Visiting Research Scholar in the Center for Intelligent Information Retrieval, University of Massachusetts, Amherst. He has participated in the NTCIR information retrieval evaluation workshops and has implemented various retrieval models. He is interested mainly in information retrieval and in general natural language processing, with occasional excursions to signal processing.

Dr. Luk is a member of the Association for Computing Machinery (ACM), BCS, and the Institute of Electrical, Information and Communication Engineers (IEICE). He is a Chartered Engineer and a Chartered Information Technology Professional.