Journal of Universal Computer Science, vol. 15, no. 13 (2009), 2566-2585
submitted: 31/10/08, accepted: 13/6/09, appeared: 1/7/09 © *J.UCS*

Online Detecting and Predicting Special Patterns over Financial Data Streams

Tao Jiang, Yucai Feng
(College of Computer Science and Technology, Huazhong University of Science & Technology, PR. China
jiangtao_guido@yahoo.com.cn, fyc@dameng.com)

Bin Zhang
(Department of Computer Science, Hengyang Normal University, PR. China
zhangbin_selina@yahoo.com.cn)

Abstract: Online detecting special patterns over financial data streams is an interesting and significant work. Existing many algorithms take it as a subsequence similarity matching problem. However, pattern detection on streaming time series is naturally expensive by this means. An efficient segmenting algorithm ONSP (**ON**line **S**egmenting and **P**runing) is proposed, which is used to find the end points of special patterns. Moreover, a novel metric distance function is introduced which more agrees with human perceptions of pattern similarity. During the process, our system presents a pattern matching algorithm to efficiently match possible emerging patterns among data streams, and a probability prediction approach to predict the possible patterns which have not emerged in the system. Experimental results show that these approaches are effective and efficient for online pattern detecting and predicting over thousands of financial data streams.
Key Words: special patterns, detecting, predicting, financial data streams
Category: H.2.8, I.5, I.2

1 Introduction

Finding special patterns over financial data streams is a very interesting and valuable work. We can make use of these patterns to help us forecast the price trend of finance time series and make correct decision. A special pattern in financial time series is a set of sequential data items collected in discrete time points, describing a meaningful price tendency of evolving data items during a period of time. Fig. 1 shows a typical pattern of Up Triangle. We can observe that the price of the end point D is higher than B's price and the prices of the end points A, C and E have a nearly same level. In other words, these end points form an Up Triangle pattern.

How to find these special patterns? A kind of technique is subsequence similarity matching. However, the technology generally use Euclidean distance (L_2-norm) [Agrawal et al. 1993] and DTW [Berndt and Clifford 1996] (Dynamic Time Warping) to measure the similarity between stream subsequence and query

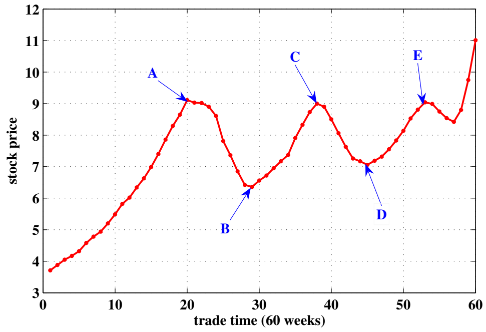


Figure 1: An Up Triangle pattern of financial data stream

subsequence. When using L_2-norm, it requires two sub sequences to keep the same length. However, for two same patterns from different financial data streams, the subsequences often keep different in length. In addition, it is well-known that Euclidean measure is very sensitive to distortion and noise [Keogh 2002]. So the effectiveness of using Euclidean distance will be very poorly. When using DTW, it often has a larger computing cost in a direct implementation and doesn't suit data stream environment in real time. Although DTW can handle local time shifting and scaling, it is still sensitive to amplitude shifting. In fact, Perng et al. (2000) refered to existing six transformations and it is especially obvious in financial data streams. So, using such methods can not obtain a good effect in detecting special patterns of financial time series.

Although existing some models can be used to detect special patterns existing in financial data streams, for example, Landmarks [Perng et al. 2000], SpADe (Spatial Assembling Distance) [Chen et al. 2007]. Landmarks model uses Landmark which can be conceived of some important data points such as extreme points to represent a data stream. The model can monitor some special patterns, for example, Double Bottom. However, its approximated function (i.e., $y = ax^3+bx^2+cx+d$) has less fidelity than linear representation (i.e., $y = ax+b$) so that the method has a lower efficiency; on the other hand, their model needs to spend much time in computing and it is difficult to adapt to the high speed data streams. Recently, Chen et al. (2007) presented SpADe model which can be used to find the patterns based-on shape for whole time series or sub sequences and define the distance of two local patterns as a weighted sum of the differences in amplitude and shape features of two local patterns. However, it is a uniform model and didn't consider the pattern features of the financial data streams.

Wu et al. (2004) pointed out that the online piecewise linear representation of financial stream data should be an up-down-up-down repetitive pattern, that is *zigzag* shape. Moreover, he also proposed an online event-driven subsequence matching algorithm over the financial data streams. They use a financial indicator %b, namely *Bollinger Band Percent*, to afford a piecewise linear representation for the stream time series. However, we observe that using %b to prune a financial stream sequence can not acquire better accuracy. Fig. 2 illustrates the problem, because the left curve and the right curve have a different trend in the graph. In addition, they define an own subsequence similarity matching function which does not suit shape pattern discovery. Because the data values of different financial streams have a very greater difference, for example, Intel stock price is 23.8$ in August 1, 2007, however IBM stock price is 112.04$ in August 1, 2007 and people generally pay more attention to the shape of stream data in pattern recognition domain.

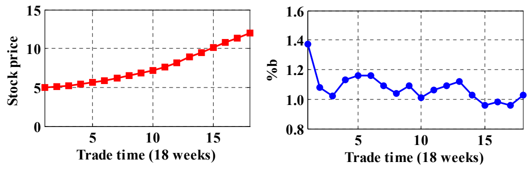


Figure 2: Left) Week close price of raw financial stream data; Right) The corresponding %b values of stream

In pattern detection on streaming time series, although we have no priori knowledge on the positions and lengths of the possible matching patterns, we can obtain feature information of some fixed pattern shape which can be utilized to segment financial stream sequence. In our paper, we make use of PLA (Piecewise Linear Approximation) [Keogh and Pazzani 1998] to segment the financial streams into a series of segments with different lengths. In the process of segmenting, we also utilize some other important feature information, for example, local extreme points and $Err(S)$ which is defined by the squared Euclidean distance between the approximated time series and the actual time series. We use a sliding window to scan financial stream and identify some critical points (namely, end points) of recent subsequences from streaming time series. Whenever a new end point is identified, we store it into a B^+-tree index and update the index. Moreover, we propose a pattern matching algorithm and a probabilistic predicting approach, and a new distance metric function which takes into account the time span and the proportion of amplitude change. Our method can efficiently identify and predict the most existing patterns of the financial time series, for example, Double Bottom, Up Triangle and Head and Shoulders, etc.

The rest of the paper is organized as follows. Section 2 briefly introduces the related works on data stream processing and pattern detecting. Section 3 describes our strategy of pre-processing over incoming streams. The algorithms of pattern matching and pattern prediction are discussed in section 4, including the pattern similarity measurement function. Section 5 is our experiment evaluation and Section 6 concludes the paper.

2 Related Work

In the literature, many research works have been proposed for stream pattern discovery. Zhu et al. (2003) studied burst detection in streams and proposed a new data structure SWT (Shifted Wavelet Tree). Bulut et al. (2005) presented a universal framework which actually is a multi-resolution indexing scheme. Y. Sakurai et al. (2007) used DTW to monitor stream patterns. C. Aggarwal (2003) proposed the concept of velocity density estimation and discussed the technique to understand, visualize and determine trends in the evolution of fast data streams. Papadimitriou and Yu (2006) introduced a method to discover optimal local patterns, which concisely describe the main trends in a time series.

However, above-mentioned works are mainly focused on a single data stream. Monitoring multiple streams is also interesting. Zhu et al. implemented a financial stream monitoring system *StatStream* [Zhu and Shasha 2002] which can be used to monitor thousands of data streams by correlation coefficients in real time. Sakurai et al. introduced BRAID [Sakurai et al 2005], which efficiently detects lag correlations between data streams. SPIRIT [Papadimitriou et al 2005] addressed the problem of capturing correlations and finding hidden variables corresponding to trends in collections of data streams. Lian et al. (2007) made use of multi-resolution approach to match data stream with a given query pattern. Recently, Zhang et al. (2007) proposed an Boolean representation based data-adaptive method for correlation analysis among a large number of streams.

To our best knowledge, there is a little of study on detecting pattern over the financial data streams, especially online monitoring special patterns, such as Double Bottom, Double Top, Bottom Triangle and UP Triangle, etc. Our work differs from previous research in several aspects. Firstly, we take a different segmenting and pruning strategy which is more suit to detect a special pattern. Secondly, a new distance metric function is used in our pattern matching approach and it takes into account not only shifting and scaling in temporal and amplitude, but also the difference of data values from different stream time series. At last, we present a probabilistic prediction which can forecast possible pattern by previous end points and historical information stored in main memory.

3 Online Stream Processing

3.1 Adjustment and smoothing

In financial market, sometimes stock price may be dramatically changed by a stock split or stock merge. So, we need to adjust the data values so that they keep on the same ratio for the change. From [Fig. 3], we can see the stock price have a sharp decrease on the position of broken line. In fact, this is caused by a stock split. For example, the close price of one stock is 20.0$ in August 17, 2007, and each share is split two shares in August 18, 2007. So, its actual price will be changed into 20.0$/2=10.0$, since then. For the case of merge, the computation is a reverse process. In our paper, we make dynamic adjustment for the financial data streams so as to keep the continuity of their data values by above method.

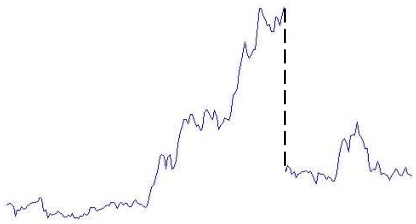


Figure 3: Stock price dramatically decreases owing to stock split

In real stock market, streaming time series often are affected by many factors which make a lot of noises existing in them. However, a pattern indicates a longer-time trend. So, smoothing is a necessary and indispensable means. An important and available way is moving average which is widely used in the financial market. In the paper, we use the method to smooth a financial data stream.

3.2 Segmenting and pruning

Before pattern matching, we need to segment financial data stream into different length segments. It actually is a process of identifying the critical points of stream sequences. A critical point is a local uppermost point or a local lowermost point. In our paper, a critical point also takes on the following characteristics: (i) it can be approximated with line curve with a smaller approximation error; (ii) it needs to undergo a longer time.

Our segmenting algorithm is based on the k-period moving average value. Similarly to the segmenting algorithm [Wu et al. 2004], we use a sliding window with varying size to arrive at the goal. But, it is not based on $\%b$ indicator. The sliding window contain at least m points and at most n points so as to guarantee the length of segment changing from m to n. It begins after the last identified end point and end right before the current point. If there are more than n points between the last end point and the current point, only the last n points are contained in the sliding window. After segmenting, a stream time series becomes an end point sequence. The following is the definition of end point.

Definition 1 *End Point: given a sliding window $W(= \{s_1, s_2, \ldots, s_w\})$ which consists of w data points from stream time series S, $m \leq w \leq n$, a threshold μ ($\mu \in \mathbb{N}$ and $\mu < w$), if $s_i \in W$ $(1 < i \leq w)$ satisfies the following conditions: (i) s_i is the maximum of W, (ii) s_i also is the maximum of set $\{s_{w+1}, \ldots, s_{w+\mu}\}$. (iii) s_i is the last one satisfying the above two condition, we call $E_i = (s_i, t_i)$ upper end point where t_i denotes the offset of current datum s_i relative to s_1. Similarly, a lower end point can be defined by a symmetric method. An upper end point or lower end point is called an end point.*

In above definition, the parameter μ is a delay time so as to confirm the local extreme value.

Pruning is a necessary refining process and it can further eliminate some shorter segments. From [Fig. 4], we can observe that there exist some shorter segments and too many end points before pruning. However, there are only five end points after pruning and the pattern of Double Bottom is more explicit.

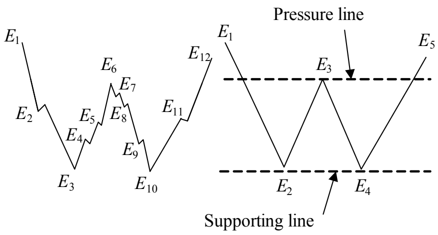


Figure 4: Left) a pattern before pruning. Right) the pattern after pruning

In the process of segmenting and pruning, we also take consideration of the following observations:

1. If the length of one segment (we denote the number of data points as the length) is less than a threshold $minLen_{seg}$, the segment would be safely removed by the segmenting and pruning process.

2. When the *approximation error* $Err(L_i)$ is larger, where L_i denotes a segment, we also can safely remove those line segments. As, for the sequence segments of a special pattern, $Err(L_i)$ often is very smaller and it will be introduced in next subsection.

3. When some conditions of a given pattern can not satisfy, we may immediately filter the pattern from the candidates and then select its next pattern to match with current pattern. For example, if end point D is small than end point B in Fig. 1, we filter the pattern from Up Triangle.

3.3 ONSP algorithm

We use linear a polynomial curve to approximate each segment of a special pattern of financial data stream by taking advantage of PLA (Piecewise Linear Approximation) technology [Keogh and Pazzani 1998]. Formally, for a given sequence segment $S = (s_1, s_2, \ldots, s_n)$ of length n, PLA can use one line segment, $y_t = a \cdot t + b$ ($t \in [1, n]$), to approximate S, where a and b are two coefficients in a linear function such that the *approximation error*, $Err_{linear}(S)$, of S is minimized. $Err_{linear}(S)$ is defined as follows.

$$Err_{linear}(S) = min(\sum_{t=1}^{n}(s_t - (a \cdot t + b))^2) \tag{1}$$

When segmenting stream time series, PLA is also computed. We approximate segment S with PLA if the *relative approximation error* $RErr_{linear}(S)$ (defined as $Err_{linear}(S)/\sum_{i=1}^{n} s_i^2$) is less than the user-defined threshold δ (i.e., $\delta = 0.02$); otherwise, we throw away the segment. Thus, we may approximate a pattern $P = (L_1, L_2, \ldots, L_N)$ with N linear segments, where L_i is the corresponding segment. However, not all segments can become the segment of a pattern. We call the segment with smaller *relative approximation error* pattern segment, whose formal definition is as follows.

Definition 2 *Pattern segment: Given a subsequence* $L(= \{s_1, s_2, \ldots, s_n\})$ *of data stream* S*,* L *is pattern segment if* $RErr_{linear}(L) \leq \delta$ *and* $n \geq minLen_{seg}$ *are satisfied, where* δ *is user defined threshold, for example,* $\delta = 0.01$*, and* $minLen_{seg}$ *is the shortest length of a pattern segment.*

Now, we can get the algorithm of segmenting and pruning, namely, ONSP (algorithm 1), which uses a sliding window policy and a data-driven mechanism to segment data streams. ONSP uses two B^+-tree indexes idx_{seg} and idx_{pat} to store current segment (from last End Point EP_{last} to current datum $datV_S$) and current End Point EP_{cur}, respectively. The detail discussion about the indexes will be introduced in [Section 4.3]. In ONSP, there are three array variables off_{FS}, off_{seg}, off_{μ}, which are used to store the offset of all streams FS (the set of all input streams), the offset of current segment, the offset of current datum $datV_S$ relative to the local extreme value $LoExV$ (maximum or minimum), respectively. The temporary variable $posLoExV$ denotes the position of $LoExV$.

In the algorithm of $Calculating\_Err$ (algorithm 2), the efficiency is very important over high speed data streams. In order to decrease the computational cost, we use a buffer $BufMA_k$ to store the last k raw data item value s_i (i is time stamp) and a variable $lastMA_k$ to store the last k-periods moving average when computing the current k-periods moving average $curMA_k$ at time stamp t. Thus, we can compute $curMA_k$ by the equation $curMA_k = (lastMA_k - BufMA_k(t-$

Algorithm 1: ONSP (FS, μ, δ, $minLen_{seg}$)

1. Initialize Array $off_{FS}[]$, $off_{seg}[]$, $off_{\mu}[]$ with 1,2,0,
   and idx_{seg} with first datums from all streams;
2. **while** (segmentation is not finished)
3.   *Receive* a new datum $datV_S$ of stream $S \in FS$;
4.   *Update* idx_{seg} with $datV_S$;
5.   **if** ($off_{seg}[S] < minLen_{seg}$) {$off_{seg}[S]$++;**continue**;}
6.   *Get* subsequence sub_S of S from idx_{seg};
     $//sub_S = S(off_{FS}[S] : off_{FS}[S] + off_{seg}[S])$
7.   **if** ($Calculating\_Err(sub_S) < \delta$)
8.   **if** ($datV_S$ is the extreme value of sub_S) {
9. off_{μ}=1, $LoExV$=$datV_S$, $posLoExV$=$off_{seg}[S]$;
10. $off_{seg}[S]$++, **continue**;}
11.   **if** ($1 \leq off_{\mu} < \mu$) $off_{\mu} = off_{\mu} + 1$;
12.   **if** ($off_{\mu} == \mu$) {
13. $off_{\mu} = 0$, $off_{FS}[S]$=$off_{FS}[S]$+$posLoExV$-1;
14.   *Update* index idx_{pat} with current *End Point* EP_{cur};
15. $posLoExV$=$off_{seg}[S]$=2;} //to next segment
16. $off_{seg}[S] = off_{seg}[S] + 1$; //to next point
17.   } **else** {$//RErr \geq \delta$
18.   **if** ($off_{\mu} > 0$) $off_{FS}[S]$=$off_{FS}[S]$+$posLoExV$-1;
19.   **if** (off_{μ}==0) $off_{FS}[S]$=$off_{FS}[S]$+$off_{seg}[S]$-2;
20.   *Update* index idx_{pat} with current *End Point* EP_{cur};
21. off_{μ}=0, $posLoExV$=$off_{seg}[S]$=2;}

$k + 1) + s_t)/k$. When computing the coefficients a, b and $RErr_{linear}$ reflected at line 2-15, we make use of an accumulative method which only needs three variables, $lastSum_{ts}$, $lastSum_s$ and $lastSum_{ss}$ to store last the corresponding aggregation values, and we can obtain the current aggregation value only by a addition operation. Then, a, b and $RErr_{linear}$ can be quickly computed by the following equation (2), (3), (4), (5), respectively,

$$a = \frac{12 * (lastSum_{ts} - \frac{(t+1)}{2} * lastSum_s)}{t * (t+1)(t-1)} \tag{2}$$

$$b = \frac{6 * (\frac{2t+1}{3} * lastSum_s - lastSum_{ts})}{t * (t-1)} \tag{3}$$

$$\begin{aligned} Err_{linear} = lastSum_{ss} + a^2 * \frac{t * (t+1)(2t+1)}{6} + a * b * t * (t+1) \\ + b^2 * t - 2a * lastSum_{ts} - 2b * lastSum_s, \end{aligned} \tag{4}$$

$$RErr_{linear} = Err_{linear}/lastSum_{ss} \tag{5}$$

Algorithm 2: Calculating_Err ($lastSum_{ts}$, $lastSum_{s}$, $lastSum_{ss}$, s_t, t)

Compute moving average $A_k(S)$ for current datum s_t;
if $(off_{seg} == minLen_{seg})$ {
  $lastSum_{ts} = lastSum_{s} = lastSum_{ss} = 0$;
  for i=1 to off_{seg} **do**
    $lastSum_{ts} = lastSum_{ts} + i * s_i$;
    $lastSum_{s} = lastSum_{s} + s_i$;
    $lastSum_{ss} = lastSum_{ss} + s_i^2$;
  end for
else
  $lastSum_{ts} = lastSum_{ts} + t * s_t$;
  $lastSum_{s} = lastSum_{s} + s_t$;
  $lastSum_{ss} = lastSum_{ss} + s_t^2$;
endif
Compute coefficients a and b by Eq. (2), (3);
Compute Err and $RErr$ by Eq. (4), (5);

where $1 \leq t \leq n$.

Through the analysis of ONSP, we can easily understand that the parameter μ and δ have a strong impact on ONSP. When using a smaller δ, all data points of *pattern segment* L_{ps} have a smaller random oscillation nearby the approximation segment L_{PLA} of L_{ps}; however, if the parameter δ is bigger, L_{PLA} will allow all data points of L_{ps} with a bigger random oscillation. Further, we also understand that the parameter μ will compact the length of L_{ps} and the number of L_{ps}. In actual processing, these limitations of ONSP lie on user choice.

4 Pattern Matching and Prediction

4.1 Pattern similarity

Our online pattern similarity matching is based on the similarity between the end points of two patterns. Although, many distance metric function have been proposed in previous years, for example, LCSS [Vlachos et al. 2002], L_p-norm and DTW, which are not suit the measurement of a financial data stream, especially in pattern detecting. Therefore, we define a novel pattern similarity metric function. Firstly, we give the definition of a pattern. Formally, a pattern is a sequence $P(= \{E_1, E_2, \ldots, E_h\})$, where E_i is the end points (upper end point or lower end point), $1 \leq i \leq h$, and h $(h \in N)$ is the number of end points. Not all patterns need to match. It is not significant to match a Head and Shoulders pattern and a Double Bottom pattern. Therefore, we must guarantee two patterns are matchable before measuring the similarity of two patterns.

Definition 3 *Matchable Pattern: For pattern* $P_1 = \{E_{11}, E_{12}, \ldots, E_{1n}\}$ *and pattern* $P_2 = \{E_{21}, E_{22}, \ldots, E_{2n}\}$, $n \in N$, P_1 *and* P_2 *are matchable patterns if they satisfy: (i)* E_{1i} *and* E_{2i} *are the same type end points (upper end point or lower end point),* $1 \leq i \leq n$*. (ii)* E_{1j} *and* $E_{1(j+1)}$ *(or* E_{2j} *and* $E_{2(j+1)}$*) are the end points of one pattern segment,* $1 \leq j \leq n-1$*. (iii)* E_{1j} *and* $E_{1(j+1)}$ *(or* E_{2j} *and* $E_{2(j+1)}$*) are the continuous end points identified by the procedure of pattern segmenting and pruning,* $1 \leq j \leq n-1$*.*

Through above definition, we can see that if P_1 and P_2 are matchable patterns, P_2 and P_3 are matchable patterns, then P_1, P_2 and P_3 are matchable patterns mutually. After giving the definition of matchable pattern, we define the distance metric function of pattern similarity as the following:

Definition 4 *Pattern Similarity: Given two patterns* $P = \{E_1, E_2, \ldots, E_h\}$ *and* $\widetilde{P} = \{\widetilde{E}_1, \widetilde{E}_2, \ldots, \widetilde{E}_h)$, P *and* $\widetilde{P}$ *are similar if the distance between pattern* P *and* $\widetilde{P}$*, that is,* $d(P, \widetilde{P})$*, satisfies the inequality* $d(P, \widetilde{P}) \leq \varepsilon$*, where* $d(P, \widetilde{P}) = \frac{1}{h-1}(\lambda_1 . \sum_{i=1}^{h-1} |A_i/B_i - \widetilde{A}_i/\widetilde{B}_i| + \lambda_2 . \sum_{i=1}^{h-1} |C_i - \widetilde{C}_i|)$, $A_i = |s_{i+1} - s_i|$, $B_i = (s_{i+1} + s_i)/2$, $C_i = (t_{i+1} - t_i)$, $\widetilde{A}_i = |\widetilde{s}_{i+1} - \widetilde{s}_i|$, $\widetilde{B}_i = (\widetilde{s}_{i+1} + \widetilde{s}_i)/2$, $\widetilde{C}_i = (\widetilde{t}_{i+1} - \widetilde{t}_i)$ *and* λ_1, λ_2 *and* $\varepsilon \geq 0$ *are user-defined parameters.*

Wu et al. (2004) defined a distance metric function which can be used to measure subsequence similarity over the financial time series. We call their function $Dist_{Wu}$. Intuitively, $Dist_{Wu}$ also can be used in pattern detecting. However, we should note that $Dist_{Wu}$ only uses absolute amplitude shifting, but not relative amplitude shifting. This makes it not suit the measurement of pattern similarity. We will explain the problem by [Fig. 5]. The prices of the corresponding end points are listed in [Tab. 1] from real stocks A, B, and C.

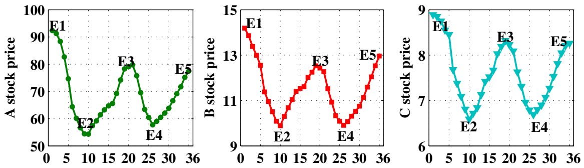


Figure 5: Three Double Bottom pattern from stock A, stock B and stock C

Now, let us to compare Wu's method and our method by computing the distance between two patterns. For simple illumination, we regard stock B as query pattern and make the parameter β in $Dist_{Wu}$ and the parameter λ_2 of $d(P, \widetilde{P})$ 0. By computing, the results of $d(B, A)$ and $d(B, C)$ are 23.165 and 1.345, respectively, using $Dist_{Wu}$. Using our approach, the results of $d(B, A)$ and $d(B, C)$ are 0.104 and 0.034, respectively. Then, we begin to compute the

Table 1: The prices of the corresponding end points from real stock A, stock B, and stock C

Stock	E_1	E_2	E_3	E_4	E_5
A	92.36	54.31	79.77	57.74	77.47
B	14.19	9.89	12.53	9.91	12.96
C	8.89	6.59	8.32	6.69	8.26

ratio of $d(B, A)/d(B, C)$. The ratio is 23.165/1.345=17.22 for the method of $Dist_{Wu}$. However, our method shows that the ratio is 0.104/0.034=3.06. This sufficiently shows our method is more adapt to pattern matching than the Wu's method.

Theorem 1. *Given two matchable patterns P_1 and P_2, the distance $d(P_1, P_2)$ is metric.*

Proof. To prove $d(P_1, P_2)$ is metric, we only need to prove it is symmetric and reflexive, and it satisfies the triangle inequality. From Definition 4, we can obviously observe that $d(P_1, P_2) \geq 0$ and $d(P_1, P_2) = d(P_2, P_1)$. So $d(P_1, P_2)$ is symmetric. Moreover, we also can find $d(P, P) = 0$. Thus, it also is reflexive. We will prove it satisfies triangle inequality as follows.

Assuming there are any three mutual matchable patterns P_1, P_2 and P_3, now we will prove the following inequality is correct:

$$d(P_1, P_3) \leq d(P_1, P_2) + d(P_2, P_3) \tag{6}$$

From Definition 4, it is obvious that the distance of two patterns are the summarization of the distances from all $h-1$ segments. Therefore, it is sufficient to give the proof on one segment. In other words, we only need to prove inequality (6) on one segment. In particular, we define the distance on the first segment as $d_1(P, \widetilde{P}) = 1/(h-1).(\lambda_1.|A_1/B_1 - \widetilde{A}_1/\widetilde{B}_1| + \lambda_2.|C_1 - \widetilde{C}_1|)$. Thus, we only need to prove the following inequality is correct:

$$d_1(P_1, P_3) \leq d_1(P_1, P_2) + d_1(P_2, P_3) \tag{7}$$

From inequality (7), we can find that $d_1(P, \widetilde{P})$ is consisted of two parts, that is, an amplitude component and a time component. According to inequality property, we have that if $a \leq b$ and $c \leq d$ $(a, b, c, d \geq 0)$, then $a + c \leq b + d$. Therefore, we only need to prove inequality (7) on each component. However, we know that if $a, b, c \geq 0$, the inequality of $|a - b| \leq |a - c| + |c - b|$ is correct. So, it will certainly be true for the following two inequalities, that is, $|A_1/B_1 - \widehat{A}_1/\widehat{B}_1| \leq |A_1/B_1 - \widetilde{A}_1/\widetilde{B}_1| + |\widetilde{A}_1/\widetilde{B}_1 - \widehat{A}_1/\widehat{B}_1|$ and $|C_1 - \widehat{C}_1| \leq |C_1 - \widetilde{C}_1| + |\widetilde{C}_1 - \widehat{C}_1|$. Since $1/(h-1)$, λ_1 and λ_2 are non-negative. Therefore, the inequality (7) is correct. Based on the inequality (7), we have the inequality (6) is correct. Now, we complete our proof.

Theorem 1 is very important because it guarantees that we can build an index to fast the similarity search and use triangular inequality filtering method.

4.2 Pattern feature

In fact, many special pattern features are be used in our algorithms. We will discuss how to make use of these special pattern features in the section. Firstly, we give the definition of a special pattern.

Definition 5 *Special Pattern: A special pattern in financial data stream especially denotes a pattern which has an explicit shape feature and a special significance, is more likely to appear again and generally need a longer-time to come into being, for example, Head and Shoulders Bottom, Double Bottom, etc.*

Given a pattern $P(=\{E_1, E_2, \ldots, E_h\})$, we can define as many features as possible. Specially, we define the pattern feature function as F_P=($P.h$, $P.trend$, $P.max$, $P.min$, ψ_1, ψ_2, ψ_3, ψ_4), which includes the main pattern features, where $P.h$ represents the number of end points, $P.trend$ denotes the whole trend (up trend and down trend), $P.max$ is the maximum of end points, $P.min$ is the minimum of end points, and ψ_1, ψ_2, ψ_3 and ψ_4 represent time shifting of two adjacent end points (that is, $t_{i+1} - t_i$), the radio of amplitude shifting (that is, $|s_{i+1} - s_i|/((s_{i+1} + s_i)/2)$), the supporting line function and the pressure line function, respectively. An example about ψ_3 and ψ_4 is shown in Fig. 4, from which we can observe that for Double Bottom pattern, the supporting line is connected two points E_2 and E_4, and the pressure line is a parallel line through E_3.

We use a feature matrix FM to store the feature information of all patterns. An example of FM is shown in [Tab. 2], where each row corresponds to a pattern, ID is identifier of a special pattern, for example, we map Double Bottom pattern to integers 1, and $Pattern$ denotes the corresponding standard sequence ($=\{E_1, E_2, \ldots, E_h\}$). FM is loaded in main memory during processing.

4.3 Pattern matching and indexing

In previous sections, we only consider pattern similarity matching on a single stream time series. In such scenario, we only store recent at most h_{max} (denotes the maximal number of end points among all patterns, i.e., $h_{max} = 10$) end points in a sliding window. Whenever a new end point is identified, we only update the window. However, in the scenario of n stream time series, for example, thousands of time series, the problem becomes more complex.

Table 2: A feature matrix for all patterns

ID	$Pattern$	h	$trend$	ε	λ_1	λ_2
1	P_1	7	$\uparrow$	0.12	1	0
2	P_2	5	$\uparrow$	0.08	1	0
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
16	P_{16}	10	$\downarrow$	0.08	1	0

At the moment, our system takes advantage of a B^+-tree to index all end points into an index idx_{pat} whose each leaf node corresponds to the entry (sid, SW_{seq}), among which sid is the id of data streams and SW_{seq} is the corresponding sliding window and stores the end points, so as to improve the efficiency of pattern matching. In fact, the sliding windows only fill up smaller the volume of main memory. Assuming there are 2000 streams and each sliding window need to h_{max}*(8+4) bytes, the total volume will be 2000*h_{max}*12=2000*10*12≈240K. So, the idx_{pat} can be loaded into main memory.

Whenever a new end point from sid data stream is identified, the system removes the oldest end point from SW_{seq} and inserts the new end point into the aftermost position of SW_{seq}. Then, a global sliding window SW_g is used to store the data of SW_{seq}. Furthermore, the system selects all possible patterns P_i from SW_g and match it with the standard pattern P_j from feature matrix FM. The detailed algorithm of pattern matching is as Algorithm 3.

Algorithm 3: Pattern Matching

Input: a feature matrix FM
Output: a set of matching patterns PS
1. Loading FM into main memory, $PS \leftarrow \emptyset$;
2. **while** a new end point E_t is identified **do** {
3. Updating the corresponding SW_{seq} of tid data stream with E_t in index tree idx_{pat};
4. $SW_g \leftarrow SW_{seq}$;
5. **for** $P_i \in SW_g$ **do**
6. **for** $P_j \in FM$ **do** {
7. **if** (P_i and P_j is not matchable) **continue**;
8. **if** $(d(P_i, P_j) \leq FM.[j].\varepsilon)$ $PS = PS \cup P_i$;
9. }
10. }// end of while

Subsequently, let's discuss the index of segmenting and pruning idx_{seg}, which is used to index the latest data points from all streams so as to find the current end points. For one stream, these indexing data items are all data points which begin after the last identified end point and end right before the current point. Similarly, the idx_{seg} is implemented by using a B^+-tree. After identifying the current end point, we can replace all data items between the last end point and the data point before the current point with future data points.

4.4 Statistical analysis and pattern prediction

In fact, we can make some statistical analysis to predict the trend precision of one type pattern. This needs to count the historical number of emergence of the type pattern and the number of hitting the corresponding trend which can be judged by the future end point and fed back. A simple method is adding two columns, that is, $freq\_P_{ID}$ and hit_trend in FM, where $freq\_P_{ID}$ denotes the emerging number of one type pattern and hit_trend represents that the pattern trend is accorded with the future trend during the next several end points.

We denote the last end point of current pattern from data stream S_i as E_{cur}, and the next l-th end point as E_l as Wu et al. (2004) noted. Here, l is an important and user defined parameter (i.e., $l = 1$ or $l = 2$). We increase hit_trend by 1 if $E_l > E_{cur} + \sigma$ (σ also is a user given parameter, $\sigma > 0$) and the *trend* is up, otherwise, if $E_l < E_{cur} - \sigma$ and the *trend* is down, we decrease 1 from hit_trend. Thus, by computing the proportion $hit\_trend/freq\_P_{ID}$ of each pattern, we can know which pattern trend is more credible.

On the other hand, we can also take advantage of the previous information of end points to predict the pattern type even if the current end point of the pattern has not still emerged. During the interval between $(h-1)^{th}$ end point and h^{th} end point, we may compute the distance of the previous $h - 1$ end points of two possible semi-patterns $d(P_{semi1}, P_{semi2})$, where P_{semi1} and P_{semi2} is consisted of the $h - 1$ end points of current data stream and feature matrix FM, respectively. Similarly, we define two variables hit_P_{ID} and $miss\_P_{ID}$ to represent hitting or missing the corresponding type pattern. If $d \leq (h-1) * \varepsilon/h$ and some critical features from feature function F_P are satisfied, we predict the current pattern of data stream as the pattern whose identifier is ID in FM. In the paper, we predict future possible pattern based on frequency and hit ratio. In other words, the predicted probability is $Prob\_P_{ID}$ which is proportional to $freq\_P_{ID}.hit\_P_{ID}/(hit\_P_{ID} + miss\_P_{ID})$. When the current end point arrives and the practical pattern is identified, we can judge if our prediction is correct. If obtaining a wrong prediction, then we feed the error back by increasing the corresponding $miss\_P_{ID}$ by $Prob\_P_{ID}$ and the corresponding $freq\_P_{ID}$ by 1. Otherwise, we increase $freq\_P_{ID}$ by 1 and hit_P_{ID} by $Prob\_P_{ID}$ for the current pattern.

5 Experiment Evaluation

5.1 Experiment setup

In order to evaluate the effectiveness and the efficiency of our proposed approaches, we make extensive experiments with real stock data. In the experiments, we use more than 1,820,000 data points of 1700 real stocks in China from `http://finance.yahoo.com/`. Among these data, about 400,000 historical data points are used to set up all parameters and build the feature matrix FM. Other 1,420,000 data points are used to test pattern similarity and pattern prediction. We conducted experiments on a Pentium 3 PC with 512 MB memory.

Before formal experiments, we require obtaining the correlative parameters. Some parameters can be gained by some experiments. Other parameter can be obtained by hand, i.e., the query patterns in FM. We use the close price of a day or a week as a financial data stream. We set up k-periods moving average with parameter $k = 3$ or 5. The size of the sliding window is changed from $m = 20$ or $m = 4$ to $n = 60$ or $n = 12$ for the time interval over days or weeks, respectively, and the threshold μ of delay time is 4 or 5 in the procedure of segmenting and pruning. The shortest length of pattern segment, $minLen_{seg}$, is 20 or 5 in the interval over days and weeks, respectively, the parameter δ about maximal relative approximation error is 0.02 on identifying pattern segment. The parameter ε, λ_1 and λ_2 are 0.2, 1 and 0, in pattern similarity, respectively. The number of end points in patterns, h_{max} is 10. In trend prediction, threshold l is 2 and σ is $\pm(10\% - 20\%)E_{cur}$.

5.2 Comparable test

In the first test, we evaluate the accuracy of several functions on pattern similarity from [Perng et al. 2000, Wu et al. 2004, Chen et al. 2007]. We call the method of Wu et al. (2004) and our method *WuPS* and *NewPS*, respectively. So, we will compare four approaches in the experiment, that is, *LandMark*, *WuPS*, *SpADe* and *NewPS*. For each type of special pattern, we pick up a representative pattern as the query pattern. There are a total of 4 query patterns. To test the accuracy of distance measures on pattern similarity, we use KNN query to retrieve the 10 nearest neighbors of each query pattern. Hits or misses of detected matching patterns are determined by the positions of pattern at the financial time series. The accuracy is evaluated based on the average error rate of the KNN queries.

As shown in [Fig. 6(a)] and [Fig. 6(b)], *NewPS* outperforms the other three methods in handling amplitude shifting and scaling. *LandMark* performs better than *WuPS* because it takes into account the proportion of amplitude change which reflects in the equation $\triangle\delta_i^{amp} = |s_i - \widetilde{s}_i|/((s_i + \widetilde{s}_i)/2)$. Note that it is

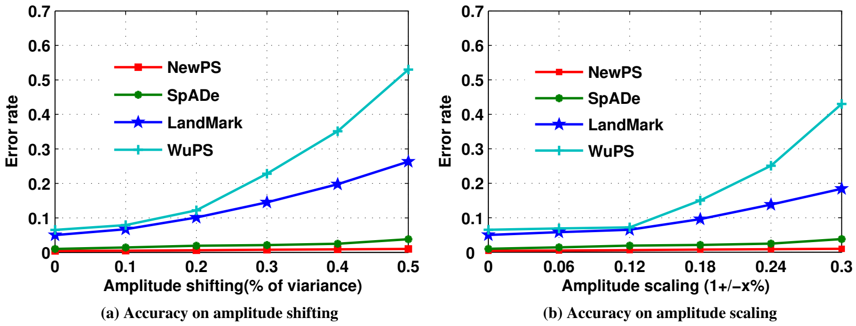


Figure 6: Accuracy comparisons under amplitude factor

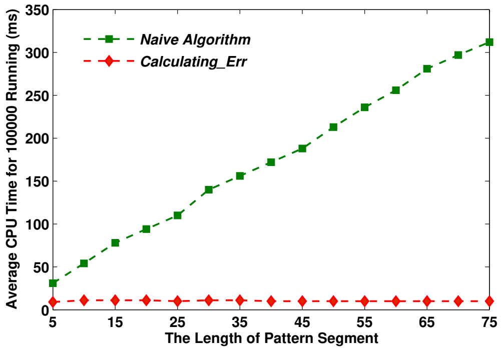


Figure 7: The comparison of CPU time for *Naive* and algorithm 2

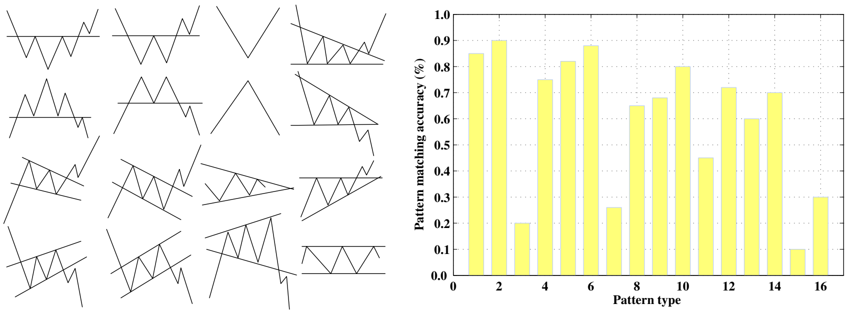
different from *NewPS*. However, *WuPS* perform worst among four functions, owing to its main consideration of subsequence similarity, but pattern similarity. Here, we omit the discussion of accuracy on pattern similarity under time shifting and scaling.

In the second test, we compare the CPU time of the *Calculating_Err* algorithm and the corresponding naive algorithm (*Naive*) which uses three circles to compute coefficients a and b, $RErr_{linear}$, respectively, for each invoking. From [Fig. 7], we observe that *Calculating_Err* is faster than *Naive*. In fact, the CPU running time of *Calculating_Err* method is almost constant and its time complexity is $O(1)$ at each sampling time t. However, the CPU running time of *Naive* gradually increases with the increase of the length of local pattern and the time complexity of *Naive* method is $O(n)$. *Calculating_Err* is 32.3 times faster than *Naive* on CPU Time when local pattern length is 75. Even if with shortest length

5, *Calculating_Err* is 3.51 times faster than *Naive* on CPU Time. But, note that the CPU time is the summary of 100,000 running owing to too short time for each invoking.

5.3 Accuracy on pattern matching

The next set of experiments study the accuracy of pattern matching algorithm. Firstly, we define the accuracy of pattern matching as the following: Pattern matching accuracy=the number of correctly matching patterns/the number of matching total patterns. Fig. (8a) shows the most familiar patterns in our pattern matching algorithm. They include Head and Shoulders pattern, V-form pattern, Triangle pattern and Banner pattern, etc.



(a) patterns (b) accuracy

Figure 8: (a) Main patterns identified by the algorithm of pattern matching and pattern prediction (b) Pattern matching accuracy vs. pattern type.

The accuracy of our algorithm reflects in [Fig. (8b)] which includes 16 type special patterns. We map integer 1, 2, ..., 16 of [Fig. (8b)] to the patterns in [Fig. (8a)] according to the order from left to right and from up to bottom. As we expected, the pattern matching algorithm attains a high accuracy for the most patterns. However, there is a smaller accuracy among V-form pattern, Scatter Triangle pattern and Rectangle pattern. In fact, it is difficult to identify above three patterns. For V-form pattern, it only includes three end points whose shape is contained in other patterns. To Rectangle pattern, we have a difficulty in determining the duration of pattern. However, to Scatter Triangle pattern, it is not easy to acquire the number of end points.

5.4 Accuracy on pattern prediction

In the section, we study pattern prediction accuracy by a series of experiments. In experiments, we extract all regular patterns from 1700 real stocks time series. In particular, the pattern prediction accuracy is measured by the recall ratio of the predicted pattern result formulated as follows: $Recall\_ratio = recall\_num/act\_num$

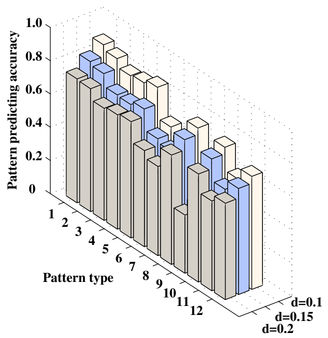


Figure 9: Pattern predicting accuracy vs. pattern type when $\varepsilon = 0.1$, $\varepsilon = 0.15$ and $\varepsilon = 0.2$

where $recall\_num$ is the number of candidates in the predicted pattern result that indeed match with the pattern in FM, and act_num is the actual number of future possible patterns that satisfy the equation $d(P_{semi1}, P_{semi2}) \leq (h-1)*\varepsilon/h$ which is introduced in [Section 4.4]. In our experiments, we only predict 12 type special pattern which didn't include V-form pattern (up form and down form), Scatter pattern and Rectangle pattern. The order of all patterns is similar to [Fig. (8b)], in addition to above four patterns. As shown in [Fig. 9], pattern predicting algorithm keeps a high predicted accuracy. For most special pattern, it achieves above 60% accuracy. Moreover, we can find that it is higher when the pattern similarity threshold is smaller. In Convergent Triangle pattern, the accuracy is lowest among all patterns, that is, 35% when $\varepsilon = 0.2$.

5.5 System performance

Finally, we evaluate the matching efficiency of the corresponding pattern in terms of the I/O cost and the CPU cost. Specifically, we compare the pattern matching on B^{+}-tree index with the no-index method. We set the page size to 1024 bytes, and run the experiments on real stock data which is consisted of the sliding window sequence SW_{seq}, including the end points of current possible patterns, from 1700 stocks. Here, we set up the length of each SW_{seq} 10.

In fact, our algorithm can be divided two situations, that is, online way and offline way. For the former, we use current quote over an interval of 1 minute, 5 minutes, 10 minutes, 30 minutes or an hour to look for the current end point. Our algorithm only spends a little time in matching and predicting a fixed pattern, due to many unqualified end points. When using the time interval over 1 minute, the average response time for a special pattern matching is only 5 milli-seconds over all 1700 stocks. It is obvious that the responding time is enough to adapt to the real-time situations. Our algorithm is up to 10 times faster than no-index method. The page accesses of the no-index method are about more 6 times than our method.

For the latter, we use the daily close price of all historical data, including more than 1,420,000 data points, as input. When using the way of batch program, our algorithm spends 21 seconds. It is 5 times faster than no-index method. On the I/O cost, the page accesses of the no-index method are about 3 times relative to our algorithm. The efficiency of the offline way is lower than the online way. This is because the offline way needs more I/O cost than the online way and it can only index a part of end points and data points of current segment into index, although it has high search efficiency on B^+-tree.

6 Conclusion

In the paper, we argue that the existing pattern matching methods do not work well in detecting pattern over multiple financial data streams when shifting and scaling exist in temporal or amplitude dimensions. We have presented a complete framework for efficiently detecting and predicting special patterns over thousands of financial data streams in an online fashion. We have demonstrated the framework is effective and efficient by extensive experiments on the large real stock datasets. In the future, we plan to evaluate the applicability of the proposed framework for the other domains, i.e., medical data and astronomical data.

References

[Aggarwal 2003] Aggarwal C. C.: "A framework for diagnosing changes in evolving data streams"; Proc. of SIGMOD Conf., ACM, San Diego, 2003, 575–586.

[Agrawal et al. 1993] Agrawal R., Faloutsos C., and Swami A.: "Efficient similarity search in sequence databases"; Proc. of the 4th Int'l Conf. on Foundations of Data Organization and Algorithms, Springer, Chicago, (1993), 69–74

[Berndt and Clifford 1996] Berndt DJ. and Clifford J.: "Finding patterns in time series: a dynamic programming approach"; Proc. of the Advances in Knowledge Discovery and Data Mining; AAAI/MIT, Menlo Park, (1996), 229–248

[Bulut and Singh 2005] Bulut A. and Singh A.: "A unified framework for monitoring data streams in real time"; Proc. of ICDE Conf., IEEE, Tokyo, (2005), 44–55.

[Chen et al. 2007] Chen Y., Nascimento M., and Ooi B. C.: "Spade: On shape-based pattern detection in streaming time series"; Proc. of ICDE Conf., IEEE, Istanbul, (2007), 786–795.

[Keogh 2002] Keogh E.: "Exact indexing of dynamic time warping"; Proc. of VLDB Conf., Morgan Kaufmann, Hong Kong, (2002), 406–417.

[Keogh and Pazzani 1998] Keogh E. and Pazzani M.: "An enhanced representation of time series which allows fast and accurate classification, clustering and relevance feedback"; Proc. of SIGKDD Conf., ACM, New York, (1998), 239–241.

[Lian et al. 2007] Lian X., Chen L., and Yu J. X.: "Similarity match over high speed time-series streams"; Proc. of ICDE Conf., IEEE, Istanbul, (2007), 1086–1095.

[Papadimitriou et al 2005] Papadimitriou S., Sun J., and Faloutsos C.: "Streaming pattern discovery in multiple time-series"; Proc. of VLDB Conf., ACM, Trondheim, (2005), 697–708.

[Papadimitriou and Yu 2006] Papadimitriou S. and Yu P. S.: "Optimal multi-scale patterns in time series streams"; Proc. of SIGMOD Conf., ACM, Chicago, (2006), 647–658.

[Perng et al. 2000] Perng C., Wang H., and Zhang S.: "Landmarks: A new model for similarity-based pattern querying in time series databases"; Proc. of ICDE Conf., IEEE, San Diego, (2000), 33–42.

[Sakurai et al. 2007] Sakurai Y., Faloutsos C., and Yamamuro M.: "Stream monitoring under the time warping distance"; Proc. of ICDE Conf., IEEE, Istanbul, (2007), 1046–1055.

[Sakurai et al 2005] Sakurai Y., Papadimitriou S., and Faloutsos C.: "Braid: Stream mining through group lag correlations"; Proc. of SIGMOD Conf., ACM, Baltimore, (2005), 599–610.

[Vlachos et al. 2002] Vlachos M., Kollios G., and Gunopulos D.: "Discovering similar multidimensional trajectories"; Proc. of ICDE, IEEE, San Jose, (2002), 673–684.

[Wu et al. 2004] Wu H., Salzberg B., and Zhang D.: "Online event-driven subsequence matching over financial data streams"; Proc. of SIGMOD Conf., ACM, Paris, (2004), 23–34.

[Zhang et al. 2007] Zhang T., Yue D., Gu Y. and Yu G.: "Boolean Representation Based Data-Adaptive Correlation Analysis over Time Series Streams"; Proc. of ACM CIKM Conf., ACM, Lisboa, (2007), 203–212.

[Zhu and Shasha 2002] Zhu Y. and Shasha D.: "Statstream: Statistical monitoring of thousands of data streams in real time"; Proc. of VLDB Conf., Morgan Kaufmann, Hong Kong, (2002), 358–369.

[Zhu and Shasha 2003] Zhu Y. and Shasha D.: "Efficient elastic burst detection in data streams"; Proc. of SIGKDD Conf., ACM, Washington, (2003), 336–345.