# Agile Prediction of Ongoing Temporal Sequences Based on Dominative Random Subsequences [*]

**Ning Yang, Changjie Tang**

*College of Computer Science, Sichuan University,*
*No.24 South Section 1, Yihuan Road,*
*Chengdu, 610065, China*
*E-mail: {yangning,cjtang}@scu.edu.cn*

Received 18 February 2012
Accepted 4 January 2013

**Abstract**

This paper identifies a new paradigm of prediction, *Agile Prediction* of ongoing temporal sequences, which achieves an acceptable accuracy just by the historical subsequences as short as possible and as close to the predicted time point as possible. To address agile prediction, a new concept, *Dominative Random Subsequence* (DRS for short), is first introduced to capture the local influence and local regularity of the subsequences that are decisive to the future of an ongoing temporal sequence. DRS mining algorithm, MDRS, and its optimal implementation OptMDRS, are also presented. In MDRS and OptMDRS, DRSs are organized as a suffix tree, DRS-Tree, to facilitate the retrieval. Next, this paper proposes an agile prediction algorithm, AgilePredict, to make accurate predictions based the DRS that is closest to the predicted time point. Finally, the results of the extensive experiments conducted on synthetic and real data sets show that our proposed method is feasible and efficient for agile prediction.

*Keywords:* Temporal sequence, Agile Prediction, Dominative Random Subsequence, Temporal similarity

## 1. Introduction

Recently, temporal sequences have been constantly emerging with different forms in a large range of applications, such as location-based service (trajectories), disease diagnosis (medical records), online recommender system (click streams) and intrusion detecting system (trace of system calls), etc. In such applications, it plays a valuable role to timely predict the coming elements of an ongoing temporal sequence with an acceptable accuracy and based on the historical data as short as possible and as close to the predicted time point as possible, since it makes these applications interact with users (or environment) more smartly and more promptly. For example, if the next location of a moving person can be predicted accurately and timely, it is possible to provide more targeted information of traffic and recommend the practically optimal route in time. In disease diagnosis, timely predicting the prognosis of disease by as few examinations as possible is always desired for the improvement of cure rate and survival rate. In intrusion detecting systems, predicting the next system call without delay by as short history of

[*]This work was supported by the National Science Foundation of China under Grant Nos. 61173099, 61103043; the Doctoral Fund of Ministry of Education of China under Grant No. 20110181120062; the Sci. & Tech. Support Programs of Sichuan Province under Grant No. 2011GZ0223.

system calls as possible is crucial to discern an attack and act promptly before the losses. At last, in online recommender systems, it is desired to predict the products of interest to a user based on his/her browsing history as short as possible.

In this paper, the prediction with an acceptable accuracy and based on the historical subsequences as short as possible and as close as possible to the predicted time point is identified as a new paradigm of prediction, ***Agile Prediction***, for its feature of *agility* which makes it different from the traditional predictions on three aspects: (1) the accuracy should be high enough to be acceptable; (2) the length of the historical subsequences should be short enough to save the computing time; (3) the distance of the historical subsequence to predicted time point should be also short enough to ensure the timeliness of prediction.

Despite a few of methodologies aiming at the prediction of temporal sequences have been proposed recent years, they often require the whole or nearly whole history of the ongoing sequence as input before an accurate prediction can be made, which makes them not always suitable for the agile prediction due to the following challenges:

- It is impractical to examine the whole history in some settings, especially under space constraints, e.g., streaming environment where only a small piece of history can be saved in memory at anytime.
- Even the whole historical data are available, to check every historical element is time-consuming.
- The whole historical data often contain noise, which leads to the risk of over-fitting.
- Agile prediction requires a tradeoff among the accuracy, the length of the historical subsequences, and the distance of the historical subsequences to the predicted time point, since longer the historical subsequences, higher the prediction accuracy, worse the agility.

This paper exploits a novel approach to overcome the above challenges. Our idea is inspired by the observation that *not all the historical time points are equally important for the future of an ongoing temporal sequence*. For example, people's life can be segmented into a various of stages in which only few stages are critical to his/her future because different choices made at these critical stages fatefully bring a different life. As another example, the future trajectory of a moving object is also dependent on few critical time points at which the positions of the object shape the future of the trajectory. So, instead of extracting the features consisting of concrete elements, like such as sequential patterns, we focus our attention on the critical time points in the history of an ongoing temporal sequence. We call the subsequence consisting of the critical time points ***Dominative Random Subsequence*** (DRS for brevity). Specifically, DRSs have the following two temporal effects that distinguish them from the ordinary subsequences:

- Local influence: The coming elements of an ongoing sequence are significantly determined by the elements occurring at the time points of the closest DRS, or in other words, the future time points are highly dependent upon the closest DRS to them.
- Local regularity: Only few of different patterns of the elements occur at a DRS, which means that the element patterns occurring at a DRS are far less random than those at other time points. In addition, higher the local regularity, shorter the DRS, which is a nice property of DRS we will prove in Section 3.

DRS is suitable for agile prediction due to its temporal effects. The local influence enables us to make an accurate prediction just based on the elements at the closest DRS instead of the whole history, while the local regularity ensures the length of DRS is moderate.

Note that a DRS is just a sequence consisting of critical time points at which the elements are influential and regular, rather than the concrete element patterns themselves. A time point can be modeled as a random variable $X$ taking values (i.e. concrete element) from an alphabet $\mathbf{\Sigma}$, and correspondingly a sequence of time points from $i$ to $j$ can be modeled as a random sequence $\boldsymbol{X_i^j} = \langle X_i, X_{i+1}, ..., X_j \rangle$. So, DRSs are essentially nothing but some particular random sequences whose influence and regularity are greater than the floor thresholds specified in advance. The following example gives a further illustration of DRS.

Table 1. Example Set of Temporal Sequences

| $X_1$ | $X_2$ | $X_3$ | $X_4$ | $X_5$ | $X_6$ | $X_7$ | $X_8$ |
|---|---|---|---|---|---|---|---|
| *f* | *x* | ***a*** | ***b*** | ***c*** | *d* | *e* | *g* |
| *a* | *u* | ***a*** | ***b*** | ***c*** | *d* | *e* | *h* |
| *q* | *x* | ***a*** | ***b*** | ***c*** | *d* | *g* | *p* |
| *f* | *u* | ***a*** | ***b*** | ***w*** | *f* | *g* | *h* |
| *b* | *n* | ***u*** | ***v*** | ***w*** | *x* | *y* | *j* |
| *c* | *r* | ***u*** | ***v*** | ***w*** | *x* | *y* | *e* |
| *c* | *t* | ***u*** | ***v*** | ***w*** | *x* | *y* | *f* |
| *b* | *n* | ***u*** | ***v*** | ***w*** | *f* | *q* | *k* |

**Example 1.** Table 1 is a training set comprising 8 temporal sequences with length of 8. It is not hard to find out that $\boldsymbol{X}_3^5$ may be a DRS because (1) the elements occurring at $\boldsymbol{X}_3^5$ is decisive to the next two time points $\boldsymbol{X}_6^7$, and (2) the number of different patterns at $\boldsymbol{X}_3^5$ is only 3, which is quite small compared with the total amount of sequences. For example, the pattern $\langle d,e \rangle$ would be much more likely to occur at $\boldsymbol{X}_6^7$ if the pattern $\langle a,b,c \rangle$ occurs at $\boldsymbol{X}_3^5$, while the pattern $\langle x,y \rangle$ is most likely when the pattern $\langle u,v,w \rangle$ occurs at $\boldsymbol{X}_3^5$. Additionally, it is obvious that every subsequence of $\boldsymbol{X}_3^5$, e.g. $\boldsymbol{X}_3^4$, is also a DRS. Hence, if we want to predict the coming elements at $\boldsymbol{X}_6^7$, we just need to check the elements at the $\boldsymbol{X}_3^5$ instead of the whole history $\boldsymbol{X}_1^5$.

In this paper, we fulfill the agile prediction of an ongoing temporal sequence through two stages. At the offline stage, the DRSs are mined from the training set of temporal sequences. At the online stage, the coming elements of an ongoing sequence are predicted just according to the elements occurring at the closest DRS.

In summary, the main contributions of this paper are as follows:

- A new temporal pattern, ***Dominative Random Subsequence***, is introduced to capture the temporal effects (i.e. the local influence and the local regularity) of temporal subsequences.
- A DRS mining algorithm, MDRS (Mining DRS), is proposed. MDRS outputs the discovered DRSs organized as a DRS-Tree, a suffix tree structure proposed by this paper to facilitate the retrieval of the DRS closest to the time point to be predicted. Additionally, to break through the performance bottleneck, the optimal implementation of MDRS, OptMDRS, is also presented.
- An agile prediction algorithm, AgilePredict, is proposed. AgilePredict is based on the DRS closest to the time point to be predicted, and thus the agility is guaranteed since the DRS is local influential and regular and with a short length relative to the length of the whole history of a temporal sequence.
- Extensive experiments conducted on synthetic and real data sets verifies the feasibility and effectiveness of our proposed method.

In the rest of this paper, we detail our methods for the discovery of DRSs and the agile prediction based on DRSs. Section 2 gives a brief review of the related work. Section 3 first investigates the measures of the influence and the regularity of a random sequence, then introduces the concepts of DRS and DRS-Tree. Section 4 presents the algorithm of mining DRSs. Section 5 proposes the agile prediction algorithm based on DRSs. Section 6 verifies our proposed methods by extensive experiments. Finally, we conclude this paper in section 7.

## 2. Related Work

This section first briefly reviews three domains relevant to our work, sequential pattern mining, sequence classification and early prediction of sequences, then points out the difference between our work and the related work.

**Sequential Pattern Mining** The goal of sequential pattern mining is to discover the frequently occurring ordered elements or subsequences as patterns [1]. There have been several typical algorithms including GSP [2], PrefixSpan [3], Spam [4], Spade [5] and SeqIndex [6]. Additionally, the sequence alignment has been investigated as the extention of sequence analysis [7, 8]. All the existing algorithms can only deal with a sequence as a whole.

**Sequence classification** Sequence classification is an extensively studied problem. Some sequence classifiers are built based on frequent sequential patterns. Karwath *et al.* [9] propose an algorithm to build the sequence classifier which takes the frequent patterns as input features. Tseng [10] proposes a Classify-By-Sequence (CBS) algorithm combining sequential pattern mining and classification. Exarchos *et al.* [11] combine sequential pattern mining and classification followed by an optimization algorithm with a higher accuracy than CBS. Some other sequence classifiers are built with artificial neural networks (ANN) [12, 13]. Wu *et al.* [12] map the sequences into vectors of $k$-gram frequencies which are used as the input of ANN. Ma *et al.* [13] use ANN to classify an UCI data set, E.Coli Promoter. In bioinformatics, some methods are proposed for the prediction of outer membrane proteins from protein sequences by the combination of the support vector machines (SVM) and feature selection [14–16].

**Early Prediction for Sequences** Alonso *et al.* [17] introduce the concept of early prediction. In [17], early prediction is achieved through the linear combination of features, which can tolerate the miss of some features but suffers the deterioration of accuracy. Xing *et al.* [18] reduce the problem of early prediction to the balance between the earliness and the accuracy of prediction, and propose a feature based method for temporal symbolic sequences, which takes a parameter $p_0$ of the expected accuracy as the input specified beforehand, and minimizes the length of the prefix that the classifier must check before a accurate prediction can be made. The method proposed by [18], however, strongly depends on the selection of concrete features. Yang *et al.* [19] investigate the kinetic regularity of information movement in early predictable sequences and utilize it as the criterion to learn the minimum number of preceding elements that are enough to make an accurate early prediction of an ongoing sequence.

Our work differs from the existing work on the following two aspects:

(1) No matter traditional classification or early prediction of sequences, they are all prefix based. Particularly, the whole sequence required by traditional sequence classification is a special prefix, i.e. the sequence itself. Although early prediction emphasizes the shortness of the sequence history, it is still prefix based since the required historical subsequence is from the very beginning element. In contrast with the existing work, agile prediction is based on the closest and shortest historical subsequence which can begin from any time point, since agile prediction emphasizes the agility.

(2) In this paper, the agile prediction is implemented by utilizing DRSs and consequently is independent of the features (patterns) consisting of concrete elements, since a DRS is just a piece of random sequence consisting of random variables which has the temporal effects of local influence and local regularity. In contrast, the existing work utilizes the concrete patterns like sequential patterns, as the features, and overlooks the temporal effects of time points, which results in the unsuitability of the existing work to agile prediction.

## 3. Dominative Random Subsequence

In this section, we first define the measures of the regularity and the influence of a random sequence, and then introduce the concepts of DRS and DRS-Tree.

### 3.1. *Regularity and Influence*

As mentioned before, a time point and a sequence of time points can be modeled as a random variable and a random sequence respectively, so we first present the measures of the regularity and the influence of a random sequence in this section. Before we get into the details, we formally define some notations. A random variable is denoted by $X$, or $X_i$ if we want to emphasize the temporal order, where the subscript is the index of time point. A random sequence is denoted by $\boldsymbol{X}_i^j$, which means $\langle X_i, X_{i+1}, \ldots, X_j \rangle$, where $i \leqslant j$. Particularly, $\boldsymbol{X}_1^n$ can be shortened as $\boldsymbol{X}^n$. Additionally, if we don't care the begin and end time points, a random sequence is also denoted by a bold capital letter such as $\boldsymbol{X}$, $\boldsymbol{Y}$. The values of $X$, $X_i$ and $\boldsymbol{X}_i^j$ are denoted by $x$, $x_i$, and $\boldsymbol{x}_i^j = \langle x_i, x_{i+1}, \ldots, x_j \rangle$ respectively. $\boldsymbol{X}_u^v \subseteq \boldsymbol{X}_i^j$ means $\boldsymbol{X}_u^v$ is a subsequence of $\boldsymbol{X}_i^j$, where $i \leqslant u \leqslant v \leqslant j$. Particularly, the set of the suffixes of $\boldsymbol{X}_i^j$ is denoted by $Suff(\boldsymbol{X}_i^j)$. The alphabet is denoted by $\mathbf{\Sigma}$, and a sequence set is denoted by $\mathbf{\Theta}$.

As mentioned earlier, the regularity of a random sequence can be intuitively evaluated by its entropy, which leads to the following definition:

**Definition 1. Regularity**. The regularity of a given random sequence $\boldsymbol{X}_i^j$, denoted by $Reg(\boldsymbol{X}_i^j)$, is defined as the reciprocal of its entropy, i.e. $Reg(\boldsymbol{X}_i^j) = 1/H(\boldsymbol{X}_i^j)$, where $H(\boldsymbol{X}_i^j)$is the entropy of $\boldsymbol{X}_i^j$.

It is obvious that less the number of different values of $\boldsymbol{X}_i^j$, less the randomness of $\boldsymbol{X}_i^j$, and greater the regularity of $\boldsymbol{X}_i^j$. Additionally, any subsequence of a regular random sequence is also regular, which is ensured by the proposition below:

**Proposition 1.** *Given a random sequence $\boldsymbol{X}_i^j$, for any random sequence $\boldsymbol{X}_u^v$ such that $\boldsymbol{X}_u^v \subseteq \boldsymbol{X}_i^j$, $Reg(\boldsymbol{X}_u^v) \geqslant Reg(\boldsymbol{X}_i^j)$. Conversely, $\boldsymbol{X}_u^v \subseteq \boldsymbol{X}_i^j$ if $Reg(\boldsymbol{X}_u^v) \geqslant Reg(\boldsymbol{X}_i^j)$ and either $[u,v] \subseteq [i,j]$ or $[i,j] \subseteq [u,v]$.*

**Proof.** Let $\boldsymbol{X} = \langle \boldsymbol{X}_i^{u-1}, \boldsymbol{X}_{v+1}^j \rangle$, then according to the chain rule of entropy [20], $H(\boldsymbol{X}_i^j) = H(\boldsymbol{X}_u^v) + H(\boldsymbol{X}|\boldsymbol{X}_u^v)$. Since the nonnegativity of entropy [20], we have $H(\boldsymbol{X}_i^j) \geqslant H(\boldsymbol{X}_u^v)$. So, by Definition 1, we get $Reg(\boldsymbol{X}_u^v) \geqslant Reg(\boldsymbol{X}_i^j)$. Conversely, if $Reg(\boldsymbol{X}_u^v) \geqslant Reg(\boldsymbol{X}_i^j)$, we have $H(\boldsymbol{X}_i^j) \geqslant H(\boldsymbol{X}_u^v)$. So, if $\boldsymbol{X}_j^j \subset \boldsymbol{X}_u^v$, then $H(\boldsymbol{X}_u^v) \geqslant H(\boldsymbol{X}_i^j)$ since conditioning reduces the entropy [20], which contradicts the known. Hence $\boldsymbol{X}_u^v \subseteq \boldsymbol{X}_i^j$ holds true. □

Note that Proposition 1 implies that greater the regularity, shorter the random sequence. So we can limit the length of random sequences by limiting the minimal regularity of them, as we will do in the next subsection.

Now we consider the measure of influence. In the context of this paper, influence means a decisive effect to the future imposed by the elements at a historical random subsequence, which results in the reduction of the uncertainty of future elements conditioned on the elements at the historical random sequence. So, it is natural to use the amount of the reduced uncertainty of the future time point to measure the influence on a future time point imposed by a random sequence , which is formulated by the following definition:

**Definition 2. Point Influence**. Given a random sequence $\boldsymbol{X}_i^j$, its influence on the future $k$th time point is defined as $PInf(\boldsymbol{X}_i^j, k) = H(X_{j+k}) - H(X_{j+k}|\boldsymbol{X}_i^j)$.

In Definition 2, the uncertainty of $X_{j+k}$ without knowing $\boldsymbol{X}_i^j$ is evaluated by the entropy $H(X_{j+k})$, while the uncertainty after knowing $\boldsymbol{X}_i^j$ is evaluated by the conditional entropy $H(X_{j+k}|\boldsymbol{X}_i^j)$. Since the conditioning reduces the entropy [20], the difference between $H(X_{j+k})$ and $H(X_{j+k}|\boldsymbol{X}_i^j)$ appropriately quantifies the reduction of the uncertainty and consequently quantifies the influence on $X_{j+k}$ imposed by $\boldsymbol{X}_i^j$. Note that the definition of influence is mathematically similar with mutual information between $X_{j+k}$ and $\boldsymbol{X}_i^j$. However, the physical implication is different. The definition of influence emphasizes the temporal order from $\boldsymbol{X}_i^j$ to $X_{j+k}$. So, $H(\boldsymbol{X}_i^j) - H(\boldsymbol{X}_i^j|X_{j+k})$, which means time can come back, is illegal and makes no sense for the measure of influence.

The overall influence on the future $k$ consecutive time points is measured by the average of the $k$ point influences, which is defined as follow:

**Definition 3. Average Influence**. Given a random sequence $\boldsymbol{X}_i^j$, its average influence on the future $k$ consecutive time points is defined as $Inf(\boldsymbol{X}_i^j, k) = \sum_{l=1}^k PInf(\boldsymbol{X}_i^j, l)/k$.

In the rest of this paper, we use ”influence” to mean the average influence by default. Intuitively, longer history has greater influence, which is exactly another important property of a random sequence as stated by Proposition 2:

**Proposition 2.** *Given random sequence $\boldsymbol{X}_i^j$, for its any suffix $\boldsymbol{X}_{i+n}^j$, $n \geqslant 0$, $Inf(\boldsymbol{X}_i^j, k) \geqslant Inf(\boldsymbol{X}_{i+n}^j, k)$ holds true for any positive integer $k$.*

**Proof.** At first, $H(X_{j+k}|\boldsymbol{X}_i^j) \leqslant H(X_{j+k}|\boldsymbol{X}_{i+n}^j)$ holds because conditional variables reduce the entropy [20]. Then according to Definition 2, we get $PInf(\boldsymbol{X}_i^j, l) \geqslant PInf(\boldsymbol{X}_{i+n}^j, l)$ for any $l \in [1,k]$, and $\sum_{l=1}^k PInf(\boldsymbol{X}_i^j, l) \geqslant \sum_{l=1}^k PInf(\boldsymbol{X}_{i+n}^j, l)$. So, according to Definition 3, $Inf(\boldsymbol{X}_i^j, k) \geqslant Inf(\boldsymbol{X}_{i+n}^j, k)$. □

### *3.2. DRS and DRS-Tree*

As introduced in Section 1, DRSs are just some sequences of critical time points of which the patterns of elements are decisive to the future elements. Before we formulate this idea in this section, it is worth noting the following two points. First, the length of DRSs should be limited to a reasonable range so as to rule out the random sequences that are trivial and meaningless for agile prediction. For example, the random sequence consisting of the whole historical time points is likely a DRS but not conducive because the agile prediction requires as few historical elements as possible. Second, ideally, we prefer the DRSs whose lengths are as short as possible and influences are as great as possible. These, however, are two conflicting objectives since Proposition 2 tells us that shorter the DRS, less the influence of it. So, it is necessary to make a tradeoff between the length of DRS and the strength of its influence. With the above considerations in mind, we get the following definition:

**Definition 4. DRS**. A random sequence $\boldsymbol{X}_i^j$ defined on a given sequence set $\boldsymbol{\Theta}$ is an $(\alpha,\beta,\gamma)$-DRS if $Reg(\boldsymbol{X}_i^j) \geqslant \gamma$ and for any $k \leqslant \alpha$, $Inf(\boldsymbol{X}_i^j,k) \geqslant \beta$ while for any $k > \alpha$, $Inf(\boldsymbol{X}_i^j,k) < \beta$, where $\alpha,\beta,\gamma$ are the floor thresholds of reach, influence and regularity respectively. All the DRSs of $\boldsymbol{\Theta}$ form a DRS set $\boldsymbol{\Pi}_\Theta(\alpha,\beta,\gamma)$.

In Definition 4, $\alpha$ is the farthest point of the future points of $\boldsymbol{X}_i^j$ at which the influence of $\boldsymbol{X}_i^j$ is not less than $\beta$. At the same time, to rule out the trivial random sequence, the length of a DRS is limited implicitly by the constraint that the regularity of $\boldsymbol{X}_i^j$ must be not less than a predefined threshold $\gamma$. Note that the first condition works because greater the regularity, shorter the random subsequence, as stated by Proposition 1.

In addition, Proposition 2 tells us that longer the suffix of a random sequence, greater the influence. So, to facilitate the discovery of the DRSs with the influence reach as far as possible under the constraints of Definition 4, we organize the DRSs of a given sequence set as a suffix tree, which is defined as follow:

**Definition 5. DRS-Tree**. The DRS-Tree of a given DRS set $\boldsymbol{\Pi}_\Theta(\alpha,\beta,\gamma)$ is a tree with the following characteristics:

(1) The root represents an empty sequence.
(2) The non-root nodes correspond one-to-one with the DRSs in $\boldsymbol{\Pi}_\Theta(\alpha,\beta,\gamma)$. Each node is labeled by a tuple ($\boldsymbol{X}_N$, $k$) where $\boldsymbol{X}_N$ is the DRS corresponding to the node $N$ and $k$ is the reach of the influence of $\boldsymbol{X}_N$.
(3) Node $A$ is the parent of Node $B$ if and only if $\boldsymbol{X}_A \in Suff(\boldsymbol{X}_B)$.

Since the nodes of a DRS-Tree correspond one-to-one with the DRSs in $\boldsymbol{\Pi}_\Theta(\alpha,\beta,\gamma)$, we also equivalently denote a DRS-Tree by $\boldsymbol{\Pi}_\Theta(\alpha,\beta,\gamma)$ in the rest of this paper. Besides, for the sake of brevity, we also represent a node of DRS-Tree by its corresponding DRS. Figure 1 shows the DRS-Tree corresponding to the DRSs set $\boldsymbol{\Pi}_\Theta(2,1,1)$ of example 1.

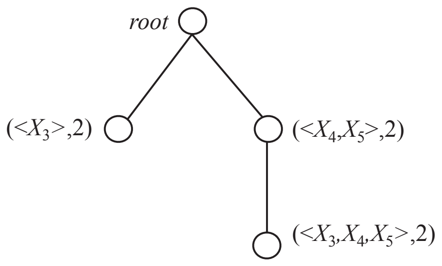

Fig. 1. The DRS-Tree of Example 1.

DRS-Tree has an important property that the DRSs corresponding to leaves are different from each other, as stated in the following proposition:

**Proposition 3.** *For any two leaves $\boldsymbol{X}_i^j$ and $\boldsymbol{X}_p^q$, $j \neq q$.*

**Proof.** If $j = q$, then either $\boldsymbol{X}_i^j \in Suff(\boldsymbol{X}_p^q)$ or $\boldsymbol{X}_p^q \in Suff(\boldsymbol{X}_i^j)$. So according to Definition 5, $\boldsymbol{X}_i^j$ is either one of the ancestors or one of the descendants of $\boldsymbol{X}_p^q$, and consequently it is impossible that they are leaves at the same time. □

## 4. Mining DRSs

**Algorithm 1:** MDRS($\boldsymbol{\Theta}, \alpha, \beta, \gamma$)

```
Input: The set of m-length sequences, $\boldsymbol{\Theta}$; The thresholds, $\alpha, \beta, \gamma$;
Output: $\boldsymbol{\Pi}_{\Theta}(\alpha, \beta, \gamma)$;
1  Initialize the candidate set
   $\boldsymbol{\Omega} = \{\langle X_i \rangle | 1 \leqslant i \leqslant m-\alpha, \text{and } Reg(\langle X_i \rangle) \geqslant \gamma\}$;
2  Initialize the root node of $\boldsymbol{\Pi}_{\Theta}(\alpha, \beta, \gamma)$;
3  while $Inf(\boldsymbol{X}, \tau) \geqslant \beta$ do
4      Pick randomly a candidate $\boldsymbol{X}$ from $\boldsymbol{\Omega}$;
5      $\tau \longleftarrow \alpha$;
6      while $Inf(\boldsymbol{X}, \tau) \geqslant \beta$ do
7          $\tau \longleftarrow \tau + 1$;
8      end
9      if $\tau > \alpha$ then
10         Add $(\boldsymbol{X}, \tau - 1)$ to $\boldsymbol{\Pi}_{\Theta}(\alpha, \beta, \gamma)$;
11     end
12     Remove $\boldsymbol{X}$ from $\boldsymbol{\Omega}$;
13     $\boldsymbol{X}' \longleftarrow$ 1-SE of $\boldsymbol{X}$;
14     if $Reg(\boldsymbol{X}') \geqslant \gamma$ then
15         Add $\boldsymbol{X}'$ to $\boldsymbol{\Omega}$;
16     end
17 end
```

This section outlines the algorithmic procedure of mining DRSs from a given temporal sequence set. The algorithm is based on the concept of *suffix extention* which is defined as follow:

**Definition 6. Suffix Extension**. Random sequence $\boldsymbol{X}$ is a *n-Suffix Extension* (shortened as *n*-SE) of random sequence $\boldsymbol{X}'$ if $\boldsymbol{X}'$ is a suffix of $\boldsymbol{X}$ and $\boldsymbol{X}$ remains *n* elements after $\boldsymbol{X}'$ is removed.

As shown in Algorithm 1, MDRS is rather straightforward. At first, it initializes the candidate set $\boldsymbol{\Omega}$ with the random sequences whose length is one and regularity is greater than the threshold $\gamma$. Then, in a while loop (line 3 to 17), each random sequence in $\boldsymbol{\Omega}$ is tested on the minimal reach threshold of the influence (line 9). If a candidate satisfies the condition, its corresponding node is added into the result DRS-Tree (line 10). Once a candidate has been checked, it is removed from the candidate set whether it is a DRS or not. After one candidate is removed, its 1-SE, however, is put into $\boldsymbol{\Omega}$ as a new candidate if the 1-SE is greater than the floor threshold of the regularity (line 14, 15). Note that when the inner while loop (line 6 to 8) terminates, the value of $\tau$ is one greater than the farthest reach of the candidates that have been checked, so the farthest reach is $\tau - 1$ when the corresponding node is added into the result (line 10).

The correctness of MDRS is guaranteed by the following proposition:

**Proposition 4.** *Given the sequence set $\boldsymbol{\Theta}$ and the floor thresholds $(\alpha, \beta, \gamma)$, MDRS discovers all the $(\alpha, \beta, \gamma)$-DRSs of $\boldsymbol{\Theta}$ when it terminates.*

**Proof.** Let $\boldsymbol{X} = \langle X_i, X_{i+1}, \ldots, X_{i+k} \rangle \in \boldsymbol{\Pi}_{\Theta}(\alpha, \beta, \gamma)$. According to the definition of DRS (Definition 3), $Reg(\boldsymbol{X}) \geqslant \gamma$. By the Proposition 1, $Reg(\langle X_{i+k} \rangle) \geqslant Reg(\boldsymbol{X})$, and consequently $Reg(\langle X_{i+k} \rangle) \geqslant \gamma$, which results in that $\langle X_{i+k} \rangle$ is an inevitable candidate (line 1). Let $\boldsymbol{S}_j$ be the *j*-SE of $\langle X_{i+k} \rangle$, where $1 \leqslant j \leqslant k$, so $\boldsymbol{X} = \boldsymbol{S}_k$. According to Proposition 1 again, we have $Reg(\langle X_{i+k} \rangle) \geqslant Reg(\boldsymbol{S}_1) \geqslant \cdots \geqslant Reg(\boldsymbol{X} = \boldsymbol{S}_k) \geqslant \gamma$, therefore $\boldsymbol{X}$ is also inevitably put into $\boldsymbol{\Omega}$ (line 14, 15). So, when $\boldsymbol{X}$ is selected from $\boldsymbol{\Omega}$ by the outer while loop, its corresponding node is inevitably added into the result (line 10) in that it is a DRS (our assumption of $\boldsymbol{X} \in \boldsymbol{\Pi}_{\Theta}(\alpha, \beta, \gamma)$). Since $\boldsymbol{X}$ represents any DRS in $\boldsymbol{\Pi}_{\Theta}(\alpha, \beta, \gamma)$, so the conclusion of the proposition holds true. ☐

Intuitively, the frequent invokes of the functions *Reg* and *Inf* will incur high computational overhead, since the computation of the entropy or conditional entropy of each suffix extension (generated at line 13) requires traversing its sequence from very beginning every time, rather than in incremental fashion. This makes the performance of MDRS is mainly dependent upon the length of sequence. In the implementation of MDRS, we utilize the *Rooted Count Tree*(shortened as RCT) [21] to compute the entropy of a random sequence. In such case, the temporal complexity of MDRS is $O(nm^3)$ where *n* is the number of sequences and *m* is the length of a sequence, as confirmed by the following proposition:

**Proposition 5.** *Given $\boldsymbol{\Theta}$ consisting of n sequences of m-length, the temporal complexity of MDRS is of $O(nm^3)$ if utilizing RCT to compute the entropy of a* *random sequence.*

**Proof.** In the worst case, every suffix extension of each random sequence will be put into the candidate set $\mathbf{\Omega}$. So the possible number of random sequences to be processed, and consequently the number of the execution times of the outer while loop (line 3-17), are both $(m-\alpha)(m-\alpha-1)/2$ (the number of edges of the complete graph where nodes consist of $X_i, i = 1, 2, \ldots, m-\alpha$). On the other hand, the temporal complexity of *Reg* and *Inf* is $O(n(m-\alpha))$ if the entropy is computed with RCT. So, the overall temporal complexity is $O(n(m-\alpha) \times (m-\alpha)(m-\alpha-1)/2) = O(nm^3)$. □

According to Proposition 5, the running time of MDRS is linear with the data size $n$ when the sequence length $m$ is fixed, and is proportional to the cube of the sequence length when the data set size is fixed. So it is clear that the efficient process of long sequence is the performance bottleneck of MDRS.

In order to break through the performance bottleneck and make the running time acceptable in practice, we implement an optimal version of MDRS, OptMDRS. If the length of sequence of $\mathbf{\Theta}$ is too long (the threshold is set to 50 in our experiments), OptMDRS will start a preprocess to split each of the sequences into shorter pieces of equal length not less than 10, before invoking MDRS. In this preprocess, the relative temporal order in the random sequences will be kept. The performance tests of MDRS and OptMDRS are detailed in Section 6.

## 5. Agile Prediction Based on DRS

As we have demonstrated, DRS is born for the agile prediction of an ongoing temporal sequence. The most significant contribution of DRS is to enable the prediction of an ongoing temporal sequence to be made only based on the historical elements occurring at the time points of the DRS that is closest to the time point being predicted, rather than all the historical elements. Here the agile prediction of an ongoing temporal sequence can be formulated as follow:

**Agile Prediction**: Given $\mathbf{\Pi}_\Theta(\alpha, \beta, \gamma)$ and the history of an ongoing temporal sequence $\boldsymbol{x}$, $\boldsymbol{x}_1^k = \langle x_1, x_2, \cdots, x_k \rangle$, computing the $\widehat{x}_{k+1}$ as the prediction of the next element $x_{k+1}$, by the history piece $\boldsymbol{x}_i^j (1 \leqslant i \leqslant j \leqslant k)$, whose length $j-i+1$ and distance to current time point $k-j$ are both as short as possible.

Our DRS based approach to fulfill the agile prediction can be outlined as follows:

(1) At first, we retrieve the DRS $\boldsymbol{X}_i^j$ $(1 \leqslant i \leqslant j \leqslant k)$ that is closest to $k$ and with shortest length (i.e. the difference $k-j$ and $j-i$ are both minimal compared with other DRSs) from the given DRS-Tree. It should be noted that only one DRS can be found since the $j$ of each branch is different from each other, as a corollary of Proposition 4.

(2) Once the DRS $\boldsymbol{X}_i^j$ is found, we examine the subsequence $\boldsymbol{x}_i^j$, i.e. a sequence of elements occurring at time points from $i$ to $j$.

(3) Then the subsequences $\{\boldsymbol{s}_i^j\}$ that are most similar to $\boldsymbol{x}_i^j$ are retrieved from the given sequence set $\mathbf{\Theta}$.

(4) Since there may be more than one instance in $\{\boldsymbol{s}_i^j\}$, the prediction $\widehat{x}_{k+1}$ is set to the $s_{k+1}$ with the highest frequency.

Here the key issue is how to measure the similarity between two temporal subsequences. Since the subsequences participating the comparison are retrieved according to the hint of the DRS, it is natural and logical to take into account the temporal property that the element at the time point closer to $k$ has a bigger say in the measure of the similarity. We call such similarity measure *Temporal Similarity*, which is defined as follow:

**Definition 7. Temporal Similarity**. The Temporal Similarity between sequences $\boldsymbol{x}_i^j$ and $\boldsymbol{s}_i^j$ is defined as $Sim(\boldsymbol{x}_i^j, \boldsymbol{s}_i^j) = \sum_{r=i}^{j} \psi(x_r, s_r)$, where $\psi(x_r, s_r)$ is defined as $\psi(x_r, s_r) = 2^{r-i}$ if $x_r = s_r$, or otherwise $\psi(x_r, s_r) = 0$.

By integrating the above ideas and the temporal similarity, the algorithm for the agile prediction of an ongoing temporal sequence is shown in Algorithm 2.

**Algorithm 2:** $\mathit{AgilePredict}(\boldsymbol{\Pi}_{\Theta}(\alpha,\beta,\gamma),\boldsymbol{x}_1^k)$

```
Input: The DRS-Tree, $\boldsymbol{\Pi}_{\Theta}(\alpha,\beta,\gamma)$; The
       history of the ongoing sequence, $\boldsymbol{x}_1^k$;
Output: $\widehat{x}_{k+1}$;
1 Search the DRS $\boldsymbol{X}_i^j$ such that $k-j$ and $j-i$
  are both minimal;
2 Search the subsequences $\{\boldsymbol{s}_i^j\}$ from $\boldsymbol{\Theta}$ such
  that $\boldsymbol{s}_i^j = \mathrm{argmax}_{\boldsymbol{s}_i^j} Sim(\boldsymbol{x}_i^j,\boldsymbol{s}_i^j)$;
3 $\widehat{x}_{k+1} \longleftarrow \mathrm{argmax}_{s_{k+1}} P(s_{k+1}|\{\boldsymbol{s}_i^j\})$
```

## 6. Experiments and Analysis

In this section, we present the experiments conducted on synthetic and real data sets to verify our methods. The experiments include three parts. At first, to validate our proposed theory about the regularity and the influence of random sequences, it is observed that how the regularity changes with the increase of the length of a random sequence and how the influence changes with the increase of the length of the history. Then, we locate the performance bottleneck of the algorithm MDRS and test the efficiency of OptMDRS. At last, we verify the agility of our proposed prediction algorithm AgilePredict by comparing with other classical algorithms for sequence prediction.

All the experiments are conducted on a PC with Intel Core I7 CPU 2.0G HZ and 4 GB main memory. The operating system is MAC OS X 10.7. All the algorithms are implemented in C with the compiler GCC 4.2.

### 6.1. *Data Sets*

**SYNTHETIC:** We generate a synthetic data set, SYN1, which is comprised of 100K sequences with length of 20. Additionally, to evaluate the performance of MDRS and OptMDRS, we also generate several other synthetic data sets of different size or with different sequence length.

**SPLICE:** SPLICE is a gene sequence data set available at UCI machine learning repository [22]. SPLICE contains 3,160 sequences and each sequence consists of 60 sequential nucleotide elements.

**BSM:** BSM is a set of system call traces recorded by an intrusion detecting system developed by MIT AI Lab [23]. BSM contains 71,760 traces and is normalized so that each trace consists of 100 system calls.

### 6.2. *Test of Regularity and Influence*

In this subsection we observe how the regularity of a random sequence changes with the increasing length and how the influence changes with the increasing length of the history.

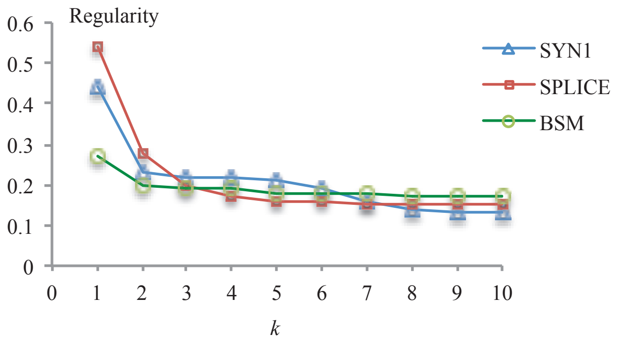

Fig. 2. $Reg(\boldsymbol{X}_1^k)(1 \leqslant k \leqslant 10)$.

Figure 2 shows the curves of regularity of the random sequence $\boldsymbol{X}_1^k$ defined on **SYN1**, **SPLICE** and **BSM** respectively, where the value of $k$ is from 1 to 10 in order. We can see from Figure 2 that the regularity is decreasing with the increase of $k$, whether $\boldsymbol{X}_1^k$ is defined on the synthetic data set or on the real data sets, as we argued in Proposition 1.

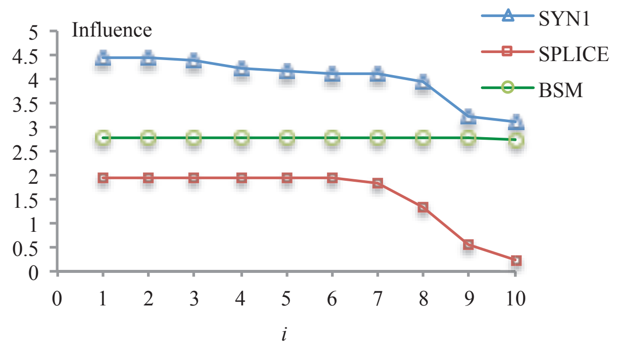

Fig. 3. $Inf(\boldsymbol{X}_i^{10},1)(1 \leqslant i \leqslant 10)$.

Figure 3 shows the influence $Inf(\boldsymbol{X}_i^{10},1)$ of the random sequence $\boldsymbol{X}_i^{10}$ defined on SYN1, SPLICE and BSM respectively, where the value of $i$ is from 1 to 10 in order. As shown in Figure 3, the influence is decreasing with the increase of $i$. Since smaller $i$ indicates shorter history, the result shown in Figure 3 verifies the conclusion of Proposition 2, i.e. longer the history, greater the influence.

## 6.3. *Performance Tests of MDRS and OptMDRS*

In this subsection, we compare the performance of the algorithm MDRS with that of the optimal implementation OptMDRS on synthetic data sets.

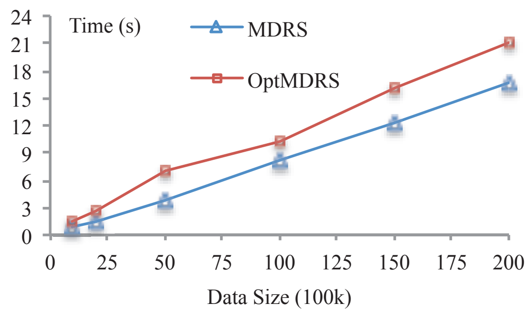

Fig. 4. Performance with increasing data set size.

We first run the MDRS and OptMDRS over the synthetic data sets of different size but with fixed sequence length 30, with the parameters of $(\alpha = 3, \beta = 0.4, \gamma = 0.2)$. As we can see from Figure 4, the running time of either MDRS or OptMDRS is approximately linear with the data size. Additionally, the running time of OptMDRS is slightly greater than that of MDRS at every data point, since OptMDRS has extra overhead incurred by the preprocess of sequence split which enlarges the size of data set.

We next run MDRS and OptMDRS over the synthetic data sets with different sequence lengths but of same size 100K, with the parameter set same as last experiment. We can find from Figure 5 that the running time of the original implementation of MDRS remarkably increases with the increase of sequence length. OptMDRS, in contrast, plays very well in that its running time is not only significantly less than MDRS's, but also linear with the length of sequence. In such case, the extra overhead the preprocess of OptMDRS incurs is negligible comparing to the performance improvement the preprocess offers.

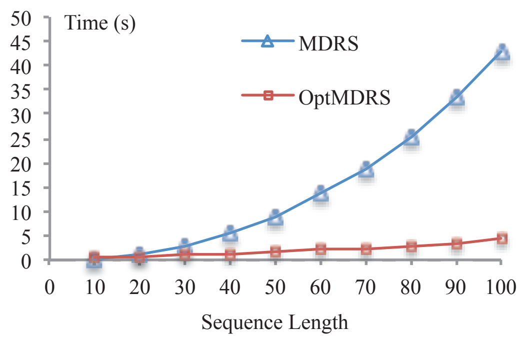

Fig. 5. Performance with increasing sequence length.

By comparing Figure 5 with Figure 4, it can be deduced that the performance of MDRS is mainly restricted by the sequence length rather than by the size of data set. This result is in line with the analysis in Section 4. It verifies that the running time of MDRS is linear with the data size $n$ when the sequence length $m$ is fixed, and is proportional to the cube of the sequence length when the data set size is fixed, as Proposition 5 states. In addition, the good performance of OptMDRS on data sets with long sequence length indicates that splitting sequences into shorter segments, as the preprocess in OptMDRS, is an effective step to break through the performance bottleneck.

## 6.4. *Tests of Agility of Prediction*

In this subsection, we compare the agility among our DRS based prediction algorithm AgilePredict, the representative early sequence prediction algorithm GSDT [18] and the classical classification algorithm ID3 [24].

As we have emphasized, the agility requires the balance of the three indexes: (1) the accuracy of prediction; (2) the length of the history based on which the prediction can be made; (3) the distance of the history to the time point to be predicted. Ideally, agile prediction of temporal sequences achieves an acceptable accuracy with the history as short as possible and as close as possible.

We conduct the experiments to obtain the above three indexes of the competitors on SYN1, BSM and SPLICE respectively. We randomly choose 80% of

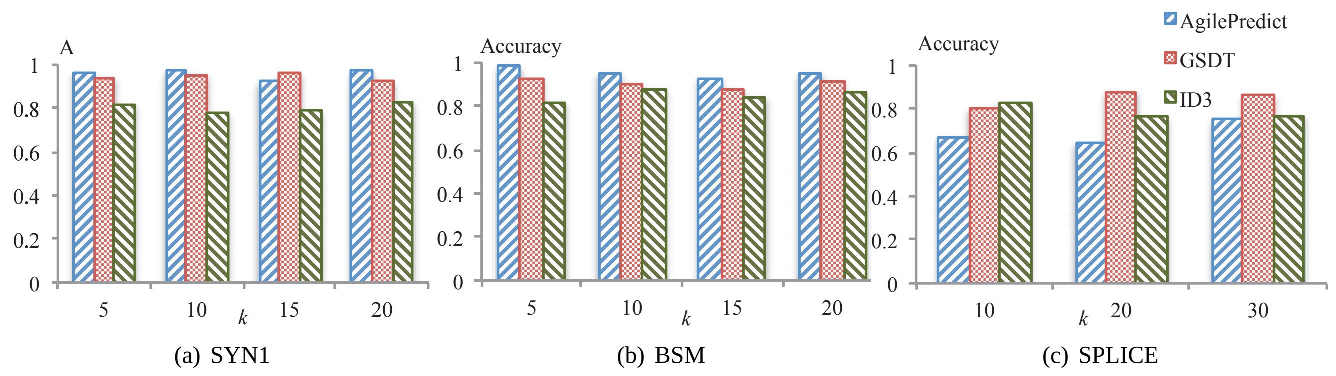

Fig. 6. Agility comparison: accuracy.

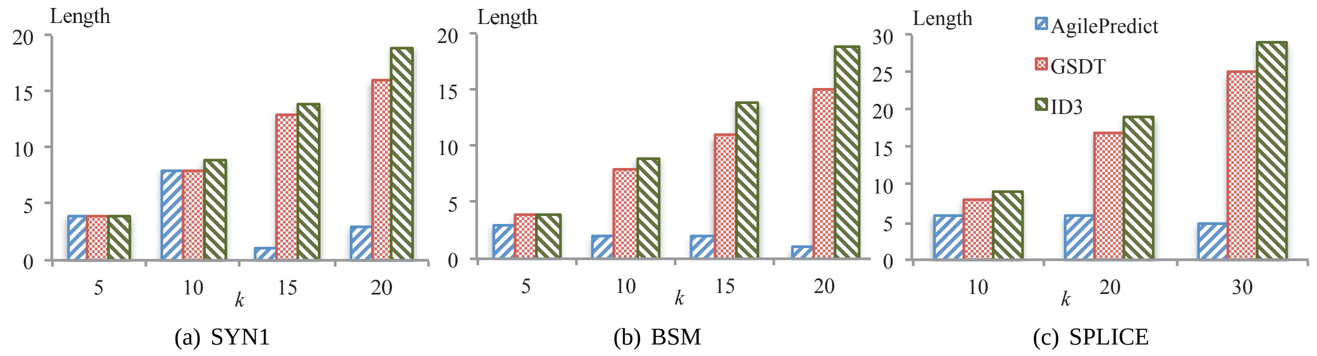

Fig. 7. Agility comparison: history length.

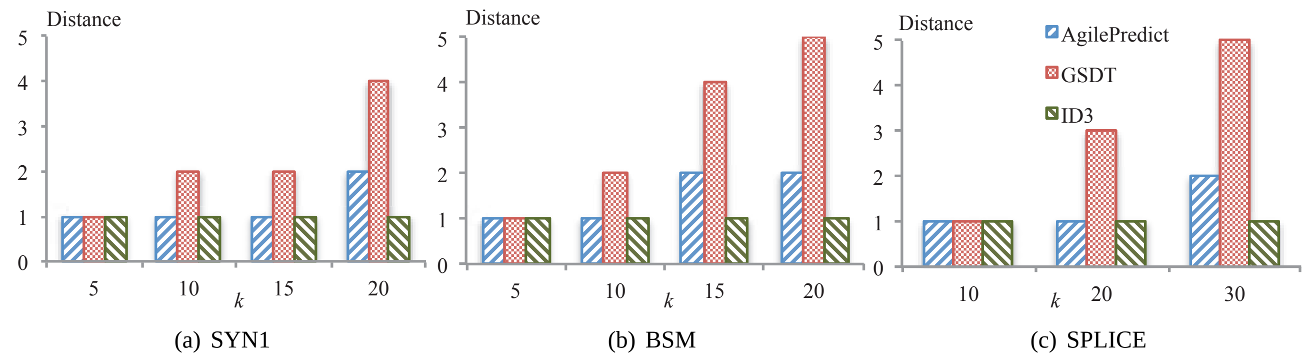

Fig. 8. Agility comparison: distance to predicted time point.

each data set as the training data and the remaining 20% as the testing data. The parameter settings are listed in the following table, where the parameters of AgilePredict are set according to the experimental results described in Subsection 6.2, and the parameters of GSDT are set as advised by [18]. Note that ID3 needs no parameters.

Table 2. Parameter settings

| | SYN1 | BSM | SPLICE |
|---|---|---|---|
| AgilePredict | $\alpha = 3$ $\beta = 0.2$ $\gamma = 0.1$ | $\alpha = 3$ $\beta = 2$ $\gamma = 0.2$ | $\alpha = 3$ $\beta = 1$ $\gamma = 0.15$ |
| GSDT | $p_0 = 0.95$ $s_0 = 0.1$ $\omega = 3$ | $p_0 = 0.95$ $s_0 = 0.1$ $\omega = 3$ | $p_0 = 0.9$ $s_0 = 0.1$ $\omega = 8$ |

### *6.4.1. Accuracy*

At first, we compare the prediction accuracies of AgilePredict, GSDT and ID3. The outcomes are shown in Figure 6, where the horizontal axis $k$ is the time point being predicted (same as the horizontal axises in Figure 7 and Figure 8).

As shown in Figure 6 (a) and (b), AgilePredict and GSDT both have high accuracies on SYN1 and BSM, no matter what value the $k$ is. By closer observation, the accuracy of AgilePredict is slightly higher than that of GSDT in most values of $k$ on SYN1 and BSM. This result suggests that an acceptable accuracy of prediction of ongoing temporal sequences can be achieved just by DRSs instead of longer history. Meanwhile, the result also suggests that our proposed measure of the temporal similarity between two sequences is practicable for the agile prediction, since it takes into account the temporal weight of different time points, i.e. the similarity at the time point closer to the time point to be predicted has higher weight than that at other time points, as mentioned in Section 5.

Figure 6 (c), however, shows that the accuracies of the tested algorithms decrease to various degrees on SPLICE, but the accuracy of AgilePredict is lower than that of GSDT, even ID3. By comparing Figure 6 (c) with (a) and (b) and considering the theoretical analysis in the preceding sections, we can conclude that the major cause of this result is that AgilePredict is good at the real temporal sequences like SYN1 and BSM, rather than gene sequences like SPLICE or sequences of other types, even the latter ones are also ordered. Particularly, our proposed measure of temporal similarity works only on the sequences that truly have the temporal property that closer the point, bigger the say in the measure of the similarity. In contrast with AgilePredict, GSDT essentially is a decision tree algorithm improved for general sequences, so GSDT is more suitable for non-temporal sequences than for temporal sequences, and AgilePredict is more competitive on temporal sequences.

At last, it can be observed that no matter which data set ID3 runs on, its accuracy is lower than AgilePredict or GSDT. This is because that ID3 was originally designed for attributed data, and consequently it is not suitable for sequence data whether it is temporal or not.

### *6.4.2. Length of history*

Figure 7 shows the lengths of the histories used by the tested algorithms to make predictions in Figure 6. We can observe from Figure 7 that the lengths of the histories used by AgilePredict are shorter than that of the histories used by the competitors. In particular, this advantage becomes more remarkable with the increase of $k$ (greater $k$ means the time point to be predicted is farther from the beginning of the sequence).

The reason is that AgilePredict is based on the historical elements occurring at the DRS whose length is indirectly restricted by the minimal threshold of the local regularity (as a result of Proposition 1), while GSDT and ID3 are prefix based, and consequently, GSDT and ID3 require longer prefixes to reach an acceptable accuracy. The result shown in Figure 7 also verifies our hypothesis introduced in Section 1, i.e. not all the time points are equally important for the future of an ongoing temporal sequence, but only few locally influential and regular segments of a sequence are critical.

### *6.4.3. Distance to predicted time point*

As mentioned before, the distance of the history to the time point to be predicted is another important factor for the success of *Agile Prediction*. Generally, shorter the distance, better the agility.

Figure 8 shows the distances of the histories used by the tested algorithms on SYN1, BSM and SPLICE respectively. We can find out that the DRSs that AgilePredict uses to make predictions are at the distances no more than 2 to various values of $k$, which is superior to the results of GSDT. Additionally, it is should be noted that although the distances of the prefixes used by ID3 are always 1, it is trivial and makes nonsense for agile prediction because ID3 always use the prefixes from the very beginning to $k-1$.

### *6.4.4. Summary*

Figure 6, Figure 7 and Figure 8 clearly show that our proposed DRS based algorithm, AgilePredict, can obtain superior accuracy on real temporal sequences just using the elements occurring at the time points of the DRS closest to the predicted time point, which indicates that AgilePredict is feasible for agile prediction of ongoing temporal sequences.

## 7. Conclusions

In this paper, we identify a new paradigm of prediction of temporal sequences, *Agile Prediction*, which makes predictions with an acceptable accuracy by the historical segments as short as possible and as close as possible to the predicted time point. To fulfill agile prediction, a new temporal concept, Dominative Random Subsequence (DRS), is proposed to capture the local temporal effects of temporal subsequences, i.e. the local influence and local regularity. Several mathematical properties of DRS and DRS's influence and regularity are investigated, which forms the foundation of our proposed agile prediction algorithm. We also present a DRS mining algorithm, MDRS, with the temporal complexity of $O(nm^3)$, and its optimal implementation OptMDRS. In MDRS and OptMDRS, DRSs are organized as a suffix tree, DRS-Tree, which facilitates the retrieval of the DRS that is closest to the time point to be predicted. A DRS based algorithm for agile prediction, AgilePredict, is proposed. AgilePredict measures the similarity between the target subsequence and the subsequences at a DRS by using our proposed measure of temporal similarity which gives the closer time point higher weight. At last, the results of the extensive experiments conducted on synthetic and real data sets verify the practicability of our proposed theory and show that our proposed algorithms are effective and efficient.

Although we made a good start in agile prediction, the problem is still far from being well solved. As part of future work, we plan to develop more accurate algorithms targeted at the temporal data with different temporal characteristics and the algorithms on numerical sequences such as continuous time series.

## References

1. G. Dong and J. Pei, "Sequence Data Mining," *Springer*, (2007).
2. M. O. N. Lesh and M. J. Zaki, "Mining features for sequence classification," *Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 342–346 (1999).
3. R. R. Srikant, "Mining sequential patterns: Generalizations and performance improvements," *Proceedings of the 5th International Conference on Extending Database Technology*, 3–17 (1996).
4. J. Pei and J. Han, "Prefixspan: Mining sequential patterns efficiently by prefix-projected pattern growth," *Proceedings of the 17th International Conference on Data Engineering*, 215–226 (2001).
5. T. J. Ayres and J. Flannick, "Sequential pattern mining using a bitmap representation," *Proceedings of the 8th International Conference on Knowledge Discovery and Data Mining*, 429–435 (2002).
6. M. J. Zaki, "Spade: An efficient algorithm for mining frequent sequences," *Machine Learning*, 42(1), 31–60 (2001).
7. J. H. Cheng and X. Yan, "Seqindex: Indexing sequences by sequential pattern analysis," *Proceedings of 2005 SIAM International Conference on Data Mining*, 601–605 (2005).
8. P. C. Parker and A. Fern, "Gradient boosting for sequence alignment," *Proceedings of the 21st National Conference on Artificial Intelligence*, 452–457 (2006).
9. A. Karwath, K. Kersting and N. Landwehr, "Boosting relational sequence alignments," *Proceedings of the 8th IEEE International Conference on Data Mining*, 857–862 (2008).
10. VSM. Tseng, "Cbs: A new classification method by using sequential patterns," *Proceedings of 2005 SIAM International Conference on Data Mining*, 596–600 (2005).
11. T. P. Exarchos and M. G. Tsipouras, "A two-stage methodology for sequence classification based on sequential pattern mining and optimization," *Data and Knowledge Engineering*, 66(3), 467–487 (2008).
12. C. Wu and M. Berry, "Neural networks for full-scale protein sequence classification: Sequence encoding with singular value decomposition," *Machine Learning*, 21(1), 177–193 (1995).
13. Q. Ma and J. T. L. Wang, "DNA sequence classication via an expectation maximization algorithm and neural networks: a case study," *IEEE Transactions on Systems, Man and Cybernetics*, 31(4), 468–475 (2001).
14. M. K. J. Park, "Prediction of protein subcellular locations by support vector machines using compositions of amino acids and amino acid pairs," *Bioinformatics*, 19(13), 1656–1663 (2003).
15. R. She and F. Chen, "Frequent-subsequence-based

prediction of outer membrane proteins. *Proceedings of the 9th International Conference on Knowledge Discovery and Data Mining*, 436–445 (2003).

16. C. S. Sonnenburg and G. Ratsch, "Learning interpretable svms for biological sequence classification," *LNCS*, 3500(2005), 389–407 (2005).
17. C. J. Alonso and J. J. Rodriguez, "Boosting interval based literals: Variable length and early classification," *Data Mining in Time Series Databases*, (2004).
18. Z. Xing, J. Pei, G. Dong and P. S. Yu, "Mining sequence classifiers for early prediction," *Proceedings of 2008 SIAM International Conference on Data Mining*, 644–655 (2008).
19. N. Yang, J. Peng, Y. Chen, and C. Tang, "Early Prediction of Temporal Sequences Based on Information Transfer," *Proceedings of 12th International Conference on Web-Age Information Management*, 542–553 (2011).
20. T. Cover, "Elements of Information Theory, 2nd Edition," *John Wiley*, (2006).
21. D. Ron, Y. Singer, and N. Tishby, "Learning Probabilistic Automata with Variable Memory Length," *Proceedings of Seventh Annual Conference on Computational Learning Theory*, 35–36 (1994).
22. A. Asuncion and D. J. Newman, "UCI machine learning repository", (2007).
23. http://www.cs.unm.edu/ immsec/data/.
24. J. R. Quinlan, "Induction of decision trees," *Machine Learning*, 1:81–106 (1986).