9-2013

Mining Effective Multi-Segment Sliding Window for Pathogen Incidence Rate Prediction

Lei Duan

Changjie Tang

Xiasong Li

Guozhu Dong
Wright State University - Main Campus, guozhu.dong@wright.edu

Xianming Wang

See next page for additional authors

Mining Effective Multi-Segment Sliding Window for Pathogen Incidence Rate Prediction

Lei Duancor, Changjie Tang, Xiaosong Li, Guozhu Dong, Xianming Wang, Jie Zuo, Min Jiang, Zhongqi Li, Yongqing Zhang

PII: S0169-023X(13)00051-7
DOI: doi: 10.1016/j.datak.2013.05.006
Reference: DATAK 1442

To appear in: *Data & Knowledge Engineering*

Please cite this article as: Lei Duancor, Changjie Tang, Xiaosong Li, Guozhu Dong, Xianming Wang, Jie Zuo, Min Jiang, Zhongqi Li, Yongqing Zhang, Mining Effective Multi-Segment Sliding Window for Pathogen Incidence Rate Prediction, *Data & Knowledge Engineering* (2013), doi: 10.1016/j.datak.2013.05.006

Mining Effective Multi-Segment Sliding Window for Pathogen Incidence Rate Prediction

Lei Duan[a,*], Changjie Tang[a,d], Xiaosong Li[b], Guozhu Dong[c], Xianming Wang[a], Jie Zuo[a], Min Jiang[b], Zhongqi Li[a], Yongqing Zhang[a]

[a] *School of Computer Science, Sichuan University, Chengdu 610065, China*
[b] *West China School of Public Health, Sichuan University, Chengdu 610041, China*
[c] *Department of Computer Science & Engineering, Wright State University, Dayton 45435, USA*
[d] *National Key Laboratory of Air Traffic Control Automation System Technology, Chengdu 610065, China*

Abstract

Pathogen incidence rate prediction, which can be considered as time series modeling, is an important task for infectious disease incidence rate prediction and for public health. This paper investigates applying a genetic computation technique, namely GEP, for pathogen incidence rate prediction. To overcome the shortcomings of traditional sliding windows in GEP based time series modeling, the paper introduces the problem of mining effective sliding window, for discovering optimal sliding windows for building accurate prediction models. To utilize the periodical characteristic of pathogen incidence rates, a multi-segment sliding window consisting of several segments from different periodical intervals is proposed and used. Since the number of such candidate windows is still very large, a heuristic method is designed for enumerating the candidate effective multi-segment sliding windows. Moreover, methods to find the optimal sliding window and then produce a mathematical model based on that window are proposed. A performance study on real-world datasets shows that the techniques are effective and efficient for pathogen incidence rate prediction.

*Corresponding author

Email addresses: leiduan@scu.edu.cn (Lei Duan), cjtang@scu.edu.cn (Changjie Tang), lixiaosong1101@126.com (Xiaosong Li), guozhu.dong@wright.edu (Guozhu Dong)

Keywords: Data Mining, Time Series Modeling, Multi-Segment Sliding Window, Pathogen Incidence Rate Prediction

1. Introduction

1.1. Pathogen Incidence Rate Prediction (PIRP)

Infectious diseases are a serious threat to the health and well-being of the citizens of the world. Effectively preventing and responding to infectious disease outbreaks is an important issue for various national governments and international organizations. To better allocate financial and medical resources on such prevention and response, it is crucial to accurately predicate the incidence rates of various infectious diseases over time.

In this paper we study the pathogen incidence rate prediction (PIRP) problem. Pathogens are infectious microbes such as viruses, bacteria, prions, or fungi, which are responsible for propagating infectious diseases in the population. Thus, solutions for the PIRP problem can be used to predict incidence rates of infectious diseases. Moreover, solutions to PIRP can be useful to other health related prediction problems such as predicting the occurrence of new virus variants and providing early warning of outbreaks of novel strains of infectious diseases.

Several traditional time series analysis methods, such as ARMA and ARIMA [1, 2, 3], and Artificial Neural Networks (ANN) [4, 5, 6], have been widely used in PIRP. However, these methods may fail to generate accurate models due to several reasons, including their inability to capture nonlinear dynamic behavior (both ARMA and ARIMA are limited by the linear basis functions), and their inability to effectively select a small number of incidence rate values at key previous time points for use as input by the prediction function. Moreover, the ANN approach has the disadvantage of producing complicated hard-to-understand prediction models.

In contrast, the genetic programming based solution proposed in this paper uses nonlinear as well as linear functions, selects a "multi-segment" sliding window (involving a small number of non-consecutive short segments of continuous time points), and produces accurate and easy-to-understand prediction models based on the time points in the window.

Specifically, our approach to solving PIRP is based on a recently developed evolutionary computation algorithm, named GEP (Gene Expression Programming) [7]. The details of GEP will be given in Section 2. The main

advantages of applying GEP to time series prediction include: (a) GEP uses evolution to perform global search to efficiently find optimal solutions. (b) GEP is well suited to learning mathematical models from numerical data automatically without the need for substantial background knowledge on the application. (c) For time series prediction, GEP can generate models to account for trends and changes. Previous studies on GEP based time series prediction have produced very good results [8, 9, 10, 11, 12].

To apply GEP to time series prediction for a given time series R, we train a mathematical model that describes the relationship between R's value for time point t and R's historical values before time t. Instead of using all historical values to build a complicated model, a sliding window containing key time points before t is used for building an accurate and yet easy-to-understand prediction model. GEP takes R's values for time points contained in the sliding window as the independent variables. In previous GEP-based time series prediction studies, GEP uses some pre-determined sliding window to develop a mathematical model. In this study, we design a method to *find* the optimal sliding window and then produce a mathematical model based on that window.

Using flexible sliding windows Selecting the optimal sliding window is important for GEP based time series modeling. The simple sliding window consisting of a series of continuous time points before the target time point of prediction, has often been used in previous GEP based time series modeling studies. The simple sliding window of size ℓ for a target time point t is the time interval $[t-\ell-1, t-1]$. (The size of a sliding window is the number of time points the window contains.) Example 1 illustrates that using such simple sliding windows may not lead to accurate prediction models, and using "multi-segment" sliding windows is more flexible and can lead to more accurate prediction models. This paper's novelty mainly lies with using GEP and "multi-segment" sliding windows for time series prediction.

Example 1. *Consider the monthly incidence rates (per thousand persons), which we refer to as time series R, of bacillary dysentery in a region, given in the following table. Let $R(t)$ denote R's value at time point t.*

Year	*Jan*	*Feb*	*Mar*	*Apr*	*May*	*Jun*	*Jul*	*Aug*	*Sep*	*Oct*	*Nov*	*Dec*
2005	0.067	0.054	0.062	0.091	0.188	0.281	0.505	0.352	0.383	0.191	0.074	0.051
2006	0.046	0.028	0.039	0.076	0.181	0.276	0.502	0.348	0.379	0.184	0.055	0.023

Figure 1 shows two approaches using two different sliding windows; (a) uses a simple sliding window of size 3*, and (b) uses a multi-segment sliding window consisting of two segments separated by a gap. For (a) we produce a model to predict* R*'s value at time point* t *using the sliding window* $W_1 = \{t-3, t-2, t-1\}$*, trained from the dataset* $D_1 = \{(R(t-3), R(t-2), R(t-1), R(t)) \mid 4 \leq t \leq 24\}$*. For (b) we produce a model to predict* R*'s value at time point* t *using a multi-segment sliding window* $W_2 = \{t-13, t-12, t-1\}$*, trained from the dataset* $D_2 = \{(R(t-13), R(t-12), R(t-1), R(t)) \mid 14 \leq t \leq 24\}$*.*

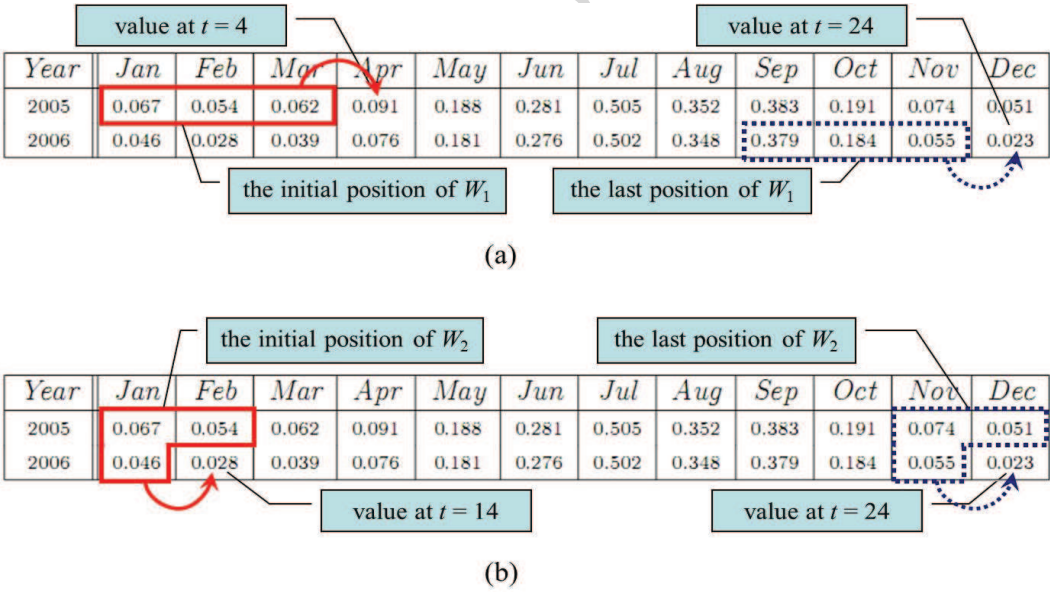


Figure 1: Time series prediction using two kinds of sliding windows

In our experiments, GEP found the following accurate model $R(t) = R(t-12)+(R(t-1)-R(t-13))*(R(t-13)/R(t-12))$ *for the multi-segment window* W_2 *but it failed to find an accurate model for the simple sliding window* W_1*.* ∎

Example 1 is synthetic. (All other examples and datasets used in this paper are from real world applications.) Example 1 demonstrates that various sliding windows can be constructed for GEP based time series modeling,

and, importantly, different sliding windows may result in different prediction accuracies – GEP may fail to develop an accurate prediction model unless suitable sliding window is constructed.

With regard to applying GEP to PIRP, it may be beneficial to take the periodicity factor into consideration. The reasons include: (a) For the pathogens of an infectious disease, the increase and decrease of incidence rates are related to certain temporal and environment factors such as season and temperature. For example, it is unreasonable to predict the incidence rates of pathogens of diarrhea in summer based on the rates in winter. (b) The inherent periodical characteristic of a pathogen gives rise to a seasonal or monthly fluctuation pattern of its incidence rates. Hence it makes sense to predict the incidence rates of a pathogen in next spring, based on its incidence rates in this spring.

1.2. Research Objectives and Contributions

This paper uses GEP-based methods to produce models for predicting incidence rates of pathogens. As the prediction accuracy of models developed by GEP depends on the sliding window, we study the new problem of mining effective sliding windows. In addition, we utilize voting in evaluating candidate sliding windows in GEP's evolution-based search of the prediction model. To the best of our knowledge, there has been no previous work on mining effective sliding window for GEP based time series modeling.

This paper makes the following four main contributions:

(a) *Problem definition for effective sliding window mining*: To overcome the shortcomings of traditional sliding windows, we introduce the problem of mining effective sliding window, for discovering optimal sliding windows for building accurate prediction model, for GEP based time series modeling.

(b) *Multi-segment sliding window*: To utilize the periodical characteristic of pathogen incidence rates, we construct a sliding window consisting of several segments from different periodical intervals. This kind of sliding window is named as multi-segment sliding window. Since the number of such candidate windows is still very large, we propose a heuristic method for enumerating candidate effective multi-segment sliding windows.

(c) *Evaluation of multi-segment sliding window*: By utilizing voting theory [13] in GEP based time series modeling, we design a method where individual genomes of GEP vote for the preferred multi-segment sliding windows in the evolution process. Based on the voting scores, the multi-segment sliding windows that are unsuitable for building accurate prediction models

are eliminated. In this way, the effective multi-segment sliding window is discovered more efficiently.

(d) *Performance evaluation*: We conduct comprehensive experiments to evaluate the performance of all proposed algorithms on real world datasets. The results indicate that the proposed methods are effective and outperform other methods for PIRP, and the methods are desirable for mining effective multi-segment sliding window and developing accurate prediction models.

1.3. Paper Outline

The remainder of this paper is organized as follows. Section 2 briefly introduces the basic concepts of GEP. Section 3 formally defines the problem of mining effective multi-segment sliding window. Section 4 presents a heuristic method for candidate effective multi-segment sliding window enumeration. Section 5 describes two GEP based methods for selecting effective multi-segment sliding windows for time series modeling. Section 6 reports an experimental study on some real world datasets. Section 7 discusses related works. Section 8 discusses future works and concluding remarks.

2. Brief Introduction to GEP

Gene Expression Programming (GEP) [7] is a recently developed variation of Genetic Algorithms and Genetic Programming for evolving with algebraic models with arbitrary form. The details of GEP are beyond the scope of this paper, we give a brief introduction below.

Both linear symbolic strings of fixed length (similar to the chromosomes of GA) and tree structures of different sizes and shapes (similar to the parse trees of GP) are used for encoding individuals (candidate solutions) in GEP, so that GEP provides new and efficient ways to program evolutionary computation [7].

As an evolutionary computation approach, the main steps of GEP are similar to those of GA and GP: (a) using populations of individuals to represent candidate solutions; (b) selecting preferred individuals based on their fitness; (c) using genetic modifications to generate new individuals of successive generations.

In GEP, the basic unit of an individual is called gene. The most distinctive feature of GEP is that each gene has access to a genotype and a corresponding phenotype: the genotype is a symbolic string of some fixed length, and the phenotype is the tree structure for the expression coded by that symbolic

string. The symbolic string of a gene is composed by two parts, a head part and a tail part, both having fixed lengths. The head part contains either function or term symbols, and the tail part contains term symbols only. The function symbol represents a mathematical operator, such as addition, subtraction, multiplication, division, log, sine. The term symbol represents an attribute value. The length of the head ($|head|$) and the length of the tail ($|tail|$) satisfy $|tail| = |head| \times (\alpha - 1) + 1$, where α is the maximum arity of the functions under consideration. The head length ($|head|$) is determined by the user as the maximum number of functions in a gene; the length of a gene ($|head| + |tail|$) remains unchanged in the middle of an execution of a given GEP algorithm. The coding region of a gene starts from the first symbol in the head, and the coding region is determined by the level based traversal (of the tree for the gene) that produces a valid arithmetic expression. As a result, despite the length of the symbolic strings of the genes is fixed, each gene can code for expression trees of different sizes and shapes.

The shortest and simplest expression of genes of a given length occurs when the first element of the string is a term, and the longest one occurs when all the symbols in the head are functions with the maximum arity (α).

The constraint between $|head|$ and $|tail|$ and the restriction that the tail only contains term symbols guarantee that each gene produces a valid algebraic expression. In GEP, an individual may involve one or more genes to encode a candidate solution. For multiple genes in an individual, the genes are connected by the linking function, such as '+'.

Suppose the function set is $\{+, -, \times, /\}$, the term set is $\{a, b, c, d, e, f\}$, the linking function is $+$. Figure 2 illustrates the expression tree and corresponding arithmetic expression of a 3-gene individual. The three genes in the individual have the same head length (3) and total length (7); but their expression trees (phenotype) are different, and so are their arithmetic expressions. For $gene1$, the coding region is the first 5 symbols (so the expression tree does not contain the "bd" at the end). For $gene2$, the coding region is the first 5 symbols (so the expression tree does not contain the "fb" at the end). For $gene3$ the coding region is the whole string. The linking function ($+$) connects the expression trees of $gene1$, $gene2$ and $gene3$ together to make up the expression tree of the individual.

The fitness function is critical for GEP algorithms since it evaluates the goodness of candidate solutions and controls the direction of evolution. The

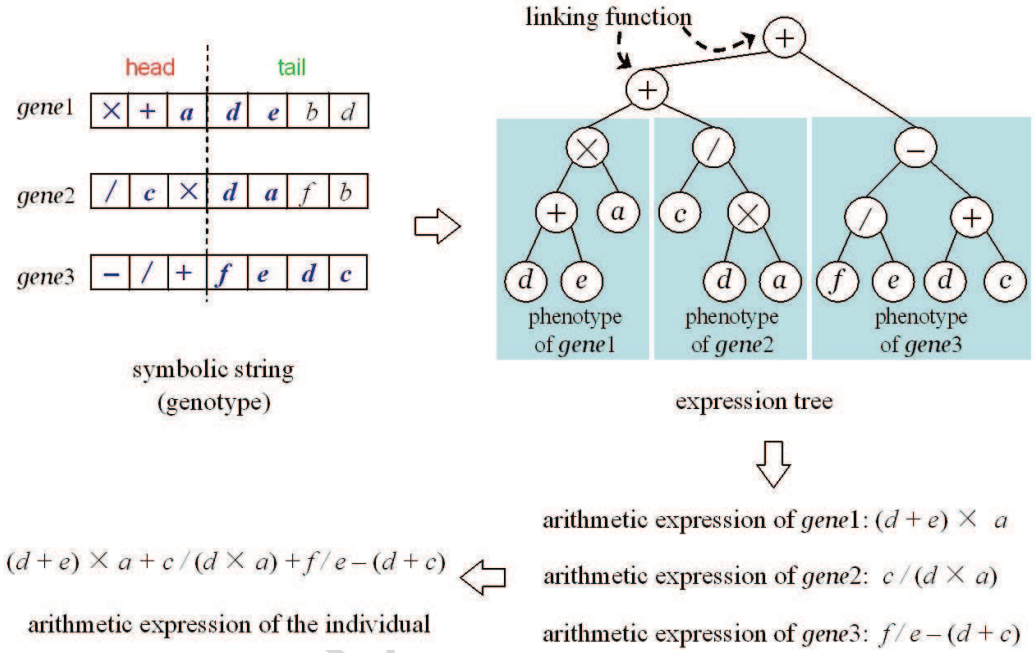


Figure 2: The genotype, phenotype and arithmetic expression of a three-gene individual

design of fitness function depends on the purpose of application. In time series modeling, the absolute error or relative error between the predicted value and the target is commonly used as fitness function.

GEP starts with a random generation of some number of individuals to make up the initial population. Based on the principle of natural selection and survival for the fittest, GEP evaluates the fitness of each individual, selects the individuals according to their fitness, and reproduces new individuals by modifying the selected individuals. Genetic modification, which creates the necessary genetic diversity, is important for GEP to eventually produce the optimal solution in the long evolutionary process.

There are three kinds of genetic modifications, namely mutation, transposition, and recombination. Mutation and transposition operate on a single individual, and recombination takes place on two individuals. A mutation can change a symbol in a gene into another symbol, as long as it does not introduce function symbols in the tail. Transposition rearranges short fragments within a gene, under some limitations. Recombination exchanges some elements between two randomly chosen individuals to form two new individu-

als. All new individuals created by GEP-style modifications are syntactically correct candidate solutions. This feature distinguishes GEP from GP, where some genetic modifications (such as mutation) can produce invalid solutions. More details can be found in [7]. The individuals of each new generation undergo the same processes of evaluation, selection and reproduction with modification as in the preceding generation. The evolution process repeats until some stop condition (given in terms of number of generations, quality of solutions, and so on) is satisfied.

Since GEP offers great potential to solve complex modeling and optimization problems, it has been used in many applications concerning symbolic regression, classification, time series analysis, cellular automata, and neural network design, etc.

3. Problem Formulation

This section introduces the basic concepts related to sliding window in GEP based PIRP and defines the problem we study in this paper.

The incidence rates of a given pathogen will be given as a time series R: $R(1), R(2), ..., R(n)$; $R(t)$ is the incidence rate at time point t for $1 \leq t \leq n$. Each t is an integer representing a time interval for incidence monitoring, such as one month. Figure 3 gives the diagram for the monthly incidence rates of bacillary dysentery (per thousand persons) over 7 years.

Sliding window: A sliding window for a target time point t is a set of time points earlier than time point t. We use $|W|$ to denote the size of a sliding window W, namely the number of time points contained in W. For example, $W_t = \{t-3, t-2, t-1\}$ is a sliding window (template) for variable time point t. When $t = 64$, $\{61, 62, 63\}$ is the sliding window instance of W_t.

In general, any set of ℓ time points that are smaller than t, can be considered a sliding window of size ℓ, provided that $\ell < t$. Thus, there are C_t^ℓ candidate sliding windows for target time point t. Even when we limit the size of sliding window ℓ to be no greater than $\frac{t}{2}$, we have the following:

$$C_t^\ell = \frac{t!}{(t-\ell)!\ell!} \geq \frac{(2\ell)!}{(2\ell-\ell)!\ell!} = \frac{(2\ell)!}{\ell!^2} \geq 2^\ell$$

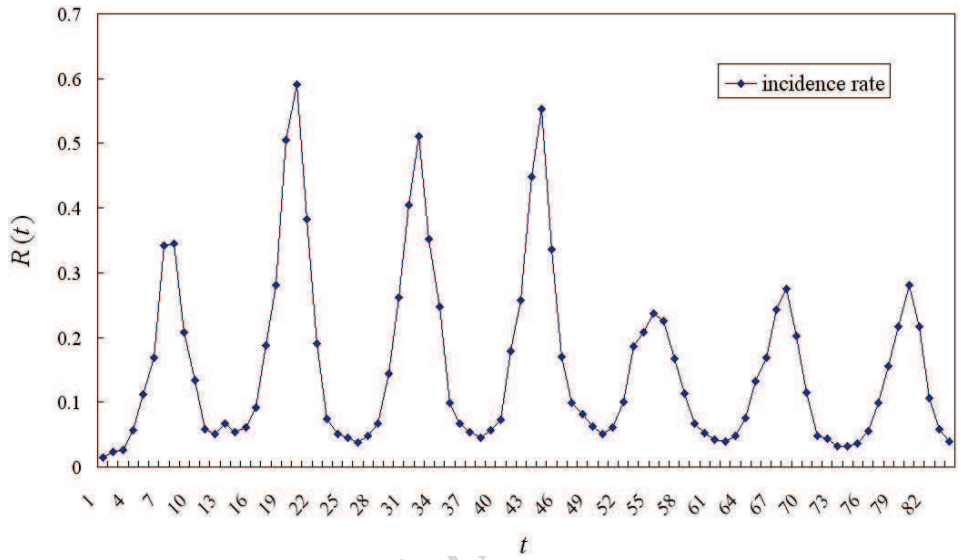


Figure 3: Monthly incidence rates of bacillary dysentery

This above implies that it is impractical to find an effective sliding window by enumerating all sliding windows. To overcome this difficulty, we utilize periodicity in sliding window construction.

Periodic partition: Due to the inherent characteristic of many pathogens, the incidence rates of a given pathogen is often periodic. Indeed, Figure 3 shows that the incidence rate of bacillary dysentery reaches its lowest point in winter, then it increases to the peak value in summer, followed by a gradual decrease to the lowest level again in winter, every year. Utilizing this periodical factor in PIRP can improve prediction accuracy.

Definition 1. (Periodic Partition) Suppose τ is a period of a given time series R. For a given time point t $(1 \leq t \leq n)$, the time points in $[1, t]$ are divided into disjoint periodic partitions, starting from t. The $|i|$-th interval $(-\lfloor \frac{t-1}{\tau} \rfloor \leq i \leq 0)$ of the periodic partition, denoted by $p_i(t)$, is $(max(0, t - (|i|+1)*\tau), t-|i|*\tau]$. The set of all partitions is called the periodic partition set of t, denoted as $P(t)$; so $P(t) = \{p_i(t) \mid -\lfloor \frac{t-1}{\tau} \rfloor \leq i \leq 0\}$.

Example 2. *For the incidence rates of bacillary dysentery (Figure 3), suppose* 12 *is a period. The corresponding periodic partitions for time point* $t =$

82 *are:* $p_0(82) : (70, 82]$, $p_{-1}(82) : (58, 70]$, $p_{-2}(82) : (46, 58]$, $p_{-3}(82) : (34, 46]$, $p_{-4}(82) : (22, 34]$, $p_{-5}(82) : (10, 22]$ *and* $p_{-6}(82) : (0, 10]$ *(see Figure 4).*

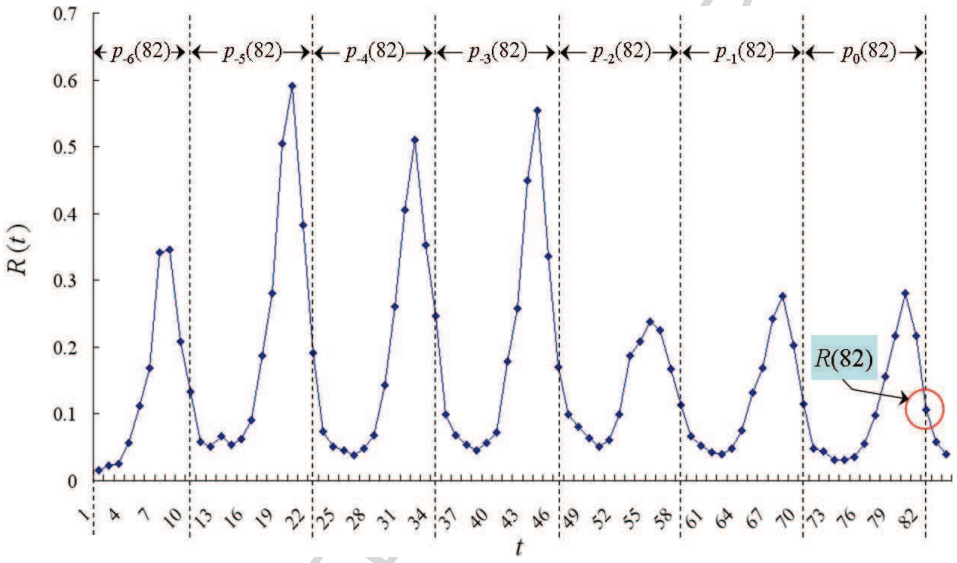


Figure 4: Periodic partitions when $t = 82$

∎

For a given periodic partition set $P(t)$, a *segment* s_i of a periodic partition $p_i(t)$ is a series of continuous time points in $p_i(t)$. The size of segment s_i, denoted by $|s_i|$, is the number of time points contained in s_i. A *size-ℓ multi-segment sliding window* W is a set of segment $\{s_i \mid p_i(t) \in P(t)\}$ satisfying (i) there is exactly one segment s_i for each $p_i(t)$ and (ii) $\sum |s_i| = \ell$.

Example 3. *For example,* $W_3 = \{58, 68, 69, 70, 80, 81\}$ *is a multi-segment sliding window for predicting* $R(82)$ *in Figure 4. The segments in* W_3 *are:* $\{\{58\}, \{68, 69, 70\}, \{80, 81\}\}$. *Observe that* $W_4 = \{52, 58, 68, 70, 74, 80\}$ *is not a multi-segment sliding window, since there are more than one segment for some periodic partitions.* ∎

The aim of this study is to find effective multi-segment sliding windows that GEP uses to build highly accurate prediction models for PIRP. Our method finds effective multi-segment sliding windows in two main steps:

(i) constructing candidate effective multi-segment sliding windows of a given size;

(ii) applying GEP to select the most effective multi-segment sliding window for building the prediction model.

4. Heuristic Enumeration of Candidate Effective Sliding Windows

It is easy to observe that the number of candidate multi-segment sliding windows is very large. So it is desirable to use some heuristic method to efficiently find some high quality candidate effective multi-segment sliding windows. From the monthly incidence rates of bacillary dysentery shown in Figure 3, we get two observations:

(i) The periodical characteristic exists in the incidence rates of bacillary dysentery. The increase and decrease trends of incidence rates in each year change in a similar manner. So, when predicting the incidence rate in a particular time interval of a given year, it is helpful to consider the incidence rates in the same time interval of previous years.

(ii) In each year, the incidence rates of bacillary dysentery increase gradually from the lowest level in January or February to the highest level in July or August. Moreover, an approximately linear increase can be found from a lowest incidence rate to the next highest one. A similar observation can be made from a highest incidence rate to the next lowest one.

Combining the two observations with the common characteristics of the incidence rates of pathogens, we design a heuristic method to enumerate candidate effective multi-segment sliding windows for PIRP.

The basic ideas of constructing a candidate effective multi-segment sliding window W for predicting the value at time point t in PIRP are: (i) The segments in W are selected from some κ periodic partitions nearest to t for some positive integer κ; we call κ the *segmentation length* of W. (ii) We pay more attention to the segments in periodic partitions closer to t than the ones further away.

Formally, for a periodic partition set $P(t)$ and an associated window W, let $S_W = \{s_i \mid s_i \subseteq p_i(t), p_i(t) \in P(t), -\kappa < i \leq 0\}$ be the set of segments

associated with W. For a given window size ℓ, W is a candidate effective size-ℓ multi-segment sliding window if S_W satisfies:

(i) $\sum_{s_i \in S_W} |s_i| = \ell$.

(ii) Except t in $p_0(t)$, there is no time point later than s_i in $p_i(t)$.

(iii) For each i, $|s_{i-1}| - |s_i| \leq \delta$, where δ is a small positive integer.

In this study, we set $\delta = 3$. Observe that condition (iii) implies that segments from periodic partitions closer to t are not much smaller than those further away. Moreover, there is no limit on how large $|s_i| - |s_{i-1}|$ is.

For a candidate effective multi-segment sliding window, the position of a segment in its corresponding periodical partition is determined (condition (ii)). Thus we can use a κ-tuple consisting of the sizes of all the segments in S_W to represent a window W.

Example 4. *Suppose the segmentation length is* 3*, and the sliding window size is* 7*. Then the candidate effective multi-segment sliding windows, denoted as triples in the form of* $< |s_{-2}|, |s_{-1}|, |s_0| >$*, are* $< 0, 0, 7 >$, $< 0, 5, 2 >$, $< 4, 1, 2 >$, $< 3, 2, 2 >$, $< 0, 4, 3 >$, $< 0, 3, 4 >$, $< 0, 2, 5 >$, $< 0, 1, 6 >$, $< 3, 0, 4 >$, $< 2, 3, 2 >$, $< 3, 1, 3 >$, $< 1, 4, 2 >$, $< 1, 1, 5 >$, $< 5, 2, 0 >$, $< 1, 2, 4 >$, $< 4, 3, 0 >$, $< 1, 0, 6 >$, $< 1, 3, 3 >$, $< 2, 1, 4 >$, $< 2, 0, 5 >$, $< 2, 2, 3 >$, $< 4, 2, 1 >$, $< 3, 3, 1 >$, $< 2, 4, 1 >$*. Once the sizes of segments are determined, the sliding window is constructed. Figure 5 illustrates the multi-segment sliding window* $< 3, 2, 2 >$ *for predicting* $R(82)$*; the corresponding segments are* $s_0 : \{79, 80, 81\}$, $s_{-1} : \{69, 70\}$, $s_{-2} : \{57, 58\}$. ∎

Proposition 1. For given sliding window size ℓ and segmentation length κ, the number of candidate effective multi-segment sliding windows is not greater than $C^{\kappa-1}_{\ell+\kappa-1}$.

Proposition 1 indicates that the number of candidate effective multi-segment sliding windows is polynomial when the segmentation length κ is fixed. Our algorithm for enumerating candidate effective multi-segment sliding windows, namely $EnumWin$, is given in Algorithm 1, which we will explain next.

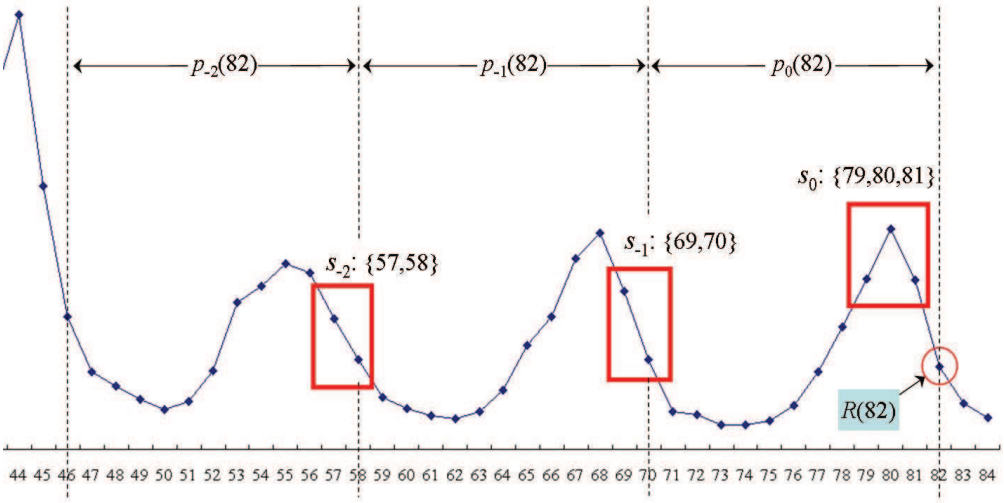


Figure 5: An example of a candidate effective multi-segment sliding window

Enumeration is started by calling $EnumWin(0, \ell, \tau - \delta - 1)$, and then called upon recursively. Observe that in each sliding window the size of s_0 is allocated at first, the total size of sliding window is ℓ, and the maximum size of s_0 is $min(\tau - 1, \ell)$. Steps 1-8 state the terminal conditions of the recursion. If the total size ℓ is allocated into κ segments satisfying the δ constraint, the κ-tuple w which records the allocated segment sizes is added to $wOut$ (Step 6). Step 9 enumerates all valid sizes of s_i. Once a κ-tuple w has been created (Step 11), a possible value is assigned to ℓ_i in w (Step 13). Step 14 enumerates possible sizes of s_{i-1} by calling $EnumWin$ recursively.

Next, we analyze the time complexity of $EnumWin$. The time complexity of finding a κ-tuple, which represents a valid candidate effective multi-segment sliding window, is $O(\kappa)$. By Proposition 1, the upper bound of time complexity of $EnumWin$ can be estimated as $O(C^{\kappa-1}_{\ell+\kappa-1} * \kappa)$. Note that, besides ℓ, the number of possible sizes of a segment depends on both $\ell_{pre} + \delta$ and τ (Step 9 in $EnumWin$). So the number of candidate effective multi-segment sliding windows would be much smaller than $C^{\kappa-1}_{\ell+\kappa-1}$, since τ is often quite small and there are few possibilities for segment sizes to be allocated when $\ell_{pre} + \delta$ is small.

Algorithm 1 EnumWin($i, \ell_{unall}, \ell_{pre}$)

Call EnumWin(0, ℓ, $\tau - \delta - 1$) to begin mining.

Input: (1) i: the subscript of segment s_i; (2) ℓ_{unall}: the size of a sliding window yet to be allocated; (3) ℓ_{pre}: the size of previous segment s_{i+1};

Output:

$wOut$: the set of κ-tuples consisting of the sizes of all segments of all individual candidate effective multi-segment sliding windows.

1: **if** $\ell_{unall} = 0$ **then**
2: $wOut \leftarrow wOut + w$;
3: return;
4: **end if**
5: **if** $|i| \geq \kappa$ **then**
6: discard w;
7: return;
8: **end if**
9: **for** $x \leftarrow 0$ to $min(\ell_{unall}, \ell_{pre} + \delta, \tau)$ **do**
10: **if** $i = 0$ **then**
11: initialize a κ-tuple w: $< \ell_{-\kappa+1}, ..., \ell_{-1}, \ell_0 >$; // ℓ_i is the size of s_i
12: **end if**
13: $w.\ell_i \leftarrow x$;
14: $EnumWin(i - 1, \ell_{unall} - x, x)$;
15: **end for**

5. Mining Effective Multi-Segment Sliding Window by GEP

Once the candidate multi-segment sliding windows have been enumerated by *EnumWin* (Algorithm 1), the next step is evaluating them and selecting the most suitable one for GEP based prediction. We propose two methods for candidate effective multi-segment sliding windows evaluation.

5.1. A Benchmark Evaluation Approach

Let R be a given time series, and W a multi-segment sliding window for variable time point t. Let $(z_1(t), ..., z_\ell(t))$ be the sequence of time points in W for t. The training dataset D associated with W is defined to be the set $\{(R(z_1(i)), ..., R(z_\ell(i)), R(i)) \mid \ell < i$ and i is a time point$\}$; so D is the set of tuples consisting of R's values for the time points in the window for time point i and R's value at time point i. A GEP individual η is a function that

takes $(R(z_1(t)), ..., R(z_\ell(t))$ as input (term set) to predict R's value at time point t.

We use the relative error between $R(t)$ and $\eta(R(z_1(t)), ..., R(z_\ell(t))$ to evaluate the η' fitness. The fitness of η on D, denoted as $fit(\eta, D)$, is computed as follows. (The pseudo-count of ϵ is used to avoid division by zero.)

$$fit(\eta, D) = \frac{1}{avg_t \frac{|\eta(R(z_1(t)),...,R(z_\ell(t))-R(t)|}{R(t)} + \epsilon}$$

Suppose DS is the set of datasets generated by all candidate effective multi-segment sliding windows associated with R.

One straightforward approach to select the effective windows is using GEP to evolve the prediction model over each dataset in DS independently. Let $D^* \in DS$ be the dataset, over which the best prediction model is evolved. Then the multi-segment sliding window which generates D^* is selected as the most effective window for GEP prediction. This straightforward approach is named as $SelectWin$, and its pseudo code is described in Algorithm 2.

In Algorithm 2, Function $CreateSeedPop(pSize)$ in Step 3 creates the initial population by generating $pSize$ individuals in a stochastic way. Function $EvaluateIndividuals(pop, D)$ in Steps 4 and 14 evaluates the fitness of each individual in pop on dataset D. If an individual with larger fitness is evolved, it is reserved as well as its associated dataset (Steps 5-9 and Steps 15-19). Function $Select(pop)$ in Step 12 selects the individuals based on the fitness to compose a new population, and Function $GeneticModify(pop)$ in Step 13 reproduces new individuals by performing genetic modifications on some selected individuals. The best individual η^* with the largest fitness is output as the prediction model. Alternatively, the initial population can be identical for each evolution; in this case, Function $CreateSeedPop(pSize)$ in Step 3 is invoked only once.

From Algorithm 2, we can see that the main routines of $SelectWin$ are similar to the basic GEP algorithm, which is desirable for solving complex modeling problem [7]. The candidate prediction models are represented as the population of individuals in $SelectWin$. The fitness function evaluates the accuracy of each candidate prediction model (Steps 4 and 14). By the selection operation, the individuals with higher fitness value are more likely to be selected for further evolution. The genetic modification (Step 13) creates the necessary diversification on candidate prediction models to generates that the final solution is globally optimal. When $SelectWin$ finishes the evolution

Algorithm 2 SelectWin($DS, pSize, ng_{max}$)

Input: (1) DS: the set of datasets generated by all candidate effective multi-segment sliding windows; (2) $pSize$: the number of individuals in a population; (3) ng_{max}: the maximum number of generations for GEP evolution;

Output:

(η^*, D^*), η^*: an individual (prediction model) with the largest fitness; D^*: the dataset that η^* most prefer.

```
1:  for each dataset D ∈ DS do
2:      ng ← 1; // ng indicates the number of evolved generations;
3:      pop ← CreateSeedPop(pSize); // initialize the population;
4:      EvaluateIndividuals(pop, D); // compute the fitness of individuals in pop;
5:      η ← GetBest(pop); // get the individual with the largest fitness in pop;
6:      if D* = ∅ or fit(η, D) > fit(η*, D*) then
7:          η* ← η;
8:          D* ← D;
9:      end if
10:     ng ← ng + 1;
11:     while ng < ng_max do
12:         pop ← Select(pop); // select individuals to compose a new population;
13:         GeneticModify(pop); // genetic modifications;
14:         EvaluateIndividuals(pop, D);
15:         η ← GetBest(pop);
16:         if fit(η, D) > fit(η*, D*) then
17:             η* ← η;
18:             D* ← D;
19:         end if
20:         ng ← ng + 1;
21:     end while
22: end for
```

on all datasets, the best prediction model is discovered as well as the optimal multi-segment sliding window.

Next, we analyze the time complexity of $SelectWin$. The evolution operations on a GEP individual include decoding, genetic operations, and evaluation. As the individual length is much less than the dataset size ($|D|$), the time complexity of operations on individuals (Steps 12-14) is $O(|pop| * |D|)$. Then for $|DS|$ datasets, the time complexity of $SelectWin$ is $O(ng_{max} * |DS| * |pop| * |D|)$.

$SelectWin$ is simple and easy to implement, but its efficiency is relatively

low, since the effective multi-segment sliding window cannot be found until the evolution of GEP on all datasets stops. In this work, we use $SelectWin$ as a benchmark algorithm and will compare it against a better one described next.

5.2. A Voting Theory based Evaluation Approach

Biological enlightenment By the biological principle known as "seek advantage, avoid disadvantage", a living being tends to develop itself in a suitable environment. GEP mimics the process of natural evolution for generating solutions to optimization problems. In PIRP, we view a prediction model under evolution as an individual, and the dataset generated by a multi-segment sliding window as an environment. The fitness of an individual is regarded as the adaption level of this individual to an environment.

Different from the benchmark approach $SelectWin$ which evolves individuals in one dataset at a time, in this subsection, we present an approach, named $VoteWin$, to involve multiple datasets in GEP evolution at the same time. In $VoteWin$, not only the individuals are evaluated on multiple datasets, but also the individuals vote for datasets generated by sliding windows (enumerated by $EnumWin$).

By evaluating the fitness of individuals on each dataset, we get the preference of the individuals for datasets. Let DS be the set of datasets generated by candidate effective multi-segment sliding windows. For an individual η, we define a partial order (called *p-order*) of η on DS to describe the datasets preference of η: $D_i \prec D_j$ if $fit(\eta, D_i) < fit(\eta, D_j)$ $(D_i, D_j \in DS, i \neq j)$. We wish to select the dataset (generated by the effective multi-segment sliding window) that most individuals prefer.

Voting for preferred dataset $VoteWin$ uses a voting method for selecting good datasets. The GEP individuals are regarded as voters and the datasets in DS are regarded as candidates. Before describing the voting method employed in $VoteWin$, we briefly introduce some basic concepts of Voting Theory. In Voting Theory, a voting method is a mapping from a set of voter preferences to an election outcome [13]. Different voting methods may give very different results. Straffin lists several fairness criterion which seem indispensable for a meaningful outcome of a voting method [14]:

- Pareto Criterion: If every voter prefers choice c_1 over choice c_2, then c_2 should not be the winner.

- Condorcet Winner Criterion: If c_1 is a choice which would win in pairwise votes against each other choice, then c_1 should be the winner.
- Condorcet Loser Criterion: If c_1 is a choice which would lose in pairwise votes against each other choice, then c_1 should not be the winner.
- Monotonicity Criterion: If choice c_1 is the winner under a voting method, and one or more voters increase their preference for c_1, then c_1 should still be the winner.

However, by Kenneth Arrow's Impossibility Theorem [15], there is no voting method that satisfies all the above fairness criteria when there are more than two candidates. Due to the impossibility of a totally fair voting method, the decision on which method to adopt should be based on what seems most fair for the situation.

If a voting method asks a voter to state a preference among candidates, it is called a preferential method. We employ two classical preferential methods [14], Borda Count and Copeland's Method, in $VoteWin$. In the Borda Count method, each voter's vote is translated into position-based points for the candidates; it selects the candidate with the most points is the winner. Copeland's Method is a voting method that elects the candidate that would win by majority rule in all pairwise comparisons; it satisfies the Condorcet Criterion. Formally, let C be the set of candidates, and V the set of voters.

- *Borda Count*: For candidate $c_i \in C$, let $rank(c_i, v_m)$ be the ranking position of c by voter $v_m \in V$. The voting score of c_i is $Borda(c_i) = \sum_{v_m \in V}(|C| - rank(c_i, v_m))$. The candidate with the largest score wins.
- *Copeland's Method*: For candidates $c_i, c_j \in C$, let $prefer(c_i, c_j, v_m) = 1$ if voter v_m prefers c_i over c_j (it is 0 otherwise). Let $count(c_i, c_j) = \sum_{v_m \in V} prefer(c_i, c_j, v_m)$ be the number of voters who prefer c_i over c_j. Let
$$win(c_i, c_j) = \begin{cases} 1 & count(c_i, c_j) > count(c_j, c_i) \\ 0 & count(c_i, c_j) = count(c_j, c_i) \\ -1 & count(c_i, c_j) < count(c_j, c_i) \end{cases}$$
The voting score of c_i is $Copeland(c_i) = \sum_{c_j \in C} win(c_i, c_j)$. The candidate with the largest score wins.

Example 5. *Suppose there are 4 candidates c_1, c_2, c_3, c_4, 4 voters v_1, v_2, v_3, v_4. The preferences of the voters are listed as follows. Then the score of*

each candidate computed by Borda Count is c_1:4, c_2:9, c_3:6, c_4:5, respectively. The score of each candidate computed by Copeland's Method is c_1:-2, c_2:3, c_3:0, c_4:-1, respectively.

voter	1*st*	2*nd*	3*rd*	4*th*	*voter*	1*st*	2*nd*	3*rd*	4*th*
v_1	c_1	c_2	c_3	c_4	v_2	c_2	c_4	c_3	c_1
v_3	c_4	c_2	c_3	c_1	v_4	c_3	c_2	c_1	c_4

Sliding window elimination By making use of voting method, $VoteWin$ eliminates "ineffective" multi-segment sliding windows. The key points are as follows.

(i) *Get the p-order of individuals.* $VoteWin$ adopts the same fitness function as $SelectWin$. The p-order of η is available once $VoteWin$ gets the fitness of η on all datasets. In GEP, the selection operation is based on fitness. The individual with higher fitness value is more likely to be selected into the next generation. For individual η, the fitness on its most preferred dataset, denoted as $fit_{select}(\eta)$, is used for selection.

(ii) *Integrate the voting score with fitness.* No matter which voting method is adopted in $VoteWin$, the voting score of each dataset is 0 in initial, and updated based on the p-order of each individual. Intuitively, the individuals with larger fitness should have more weight on voting. $VoteWin$ takes the individual's fitness into consideration when computing the voting score of each dataset. Let η^* be the best individual in *pop*, $D_i, D_j \in DS$. Then, the methods of integrating the voting score with the fitness are listed as follows:

$Borda(D_i) = \sum_{\eta \in pop} \frac{fit_{select}(\eta)}{fit_{select}(\eta^*)} * (|DS| - rank(D_i, \eta))$, if Borda Count is used.

$count(D_i, D_j) = \sum_{\eta \in pop} \frac{fit_{select}(\eta)}{fit_{select}(\eta^*)} * prefer(D_i, D_j, \eta)$, if Copeland's Method is used.

(iii) *Eliminate the sliding window with the lowest voting score one by one.* $VoteWin$ takes all datasets (generated by candidate effective multi-segment sliding windows) in DS as candidates. The voting score of each dataset is updated as the individuals evolve. Suppose the maximum number of generations for GEP evolution is ng_{max}. $VoteWin$ eliminates the sliding window with the lowest voting score every $\lfloor \frac{ng_{max}}{|DS|} \rfloor$ generations until only one dataset is left. During the evolution, the individual with the largest fitness is cloned to the next generation to guarantee that the best solution is never lost. In $VoteWin$, the fitness of an individual is associated with a dataset. If

the dataset with the lowest voting score is also the one most preferred by η^*, $VoteWin$ eliminates the dataset with the second lowest voting score instead. After a dataset is removed from DS, $VoteWin$ resets the voting scores of datasets for a new round of voting.

Like $SelectWin$, $VoteWin$ also keeps the main routines of basic GEP algorithm to evolve the prediction models. Different from $SelectWin$ which evaluates the fitness of candidate prediction models in one dataset at a time, $VoteWin$ evaluates the fitness of candidate prediction models in multiple datasets in the mean time, and uses evolution to find the optimal dataset and to evolve a prediction model based on the dataset.

Algorithm 3 describes the pseudo code of $VoteWin$. The upper bound of time complexity of $VoteWin$ is $O(ng_{max} * |pop| * (|DS| * |D| + |DS|^2))$. However, $VoteWin$ is more efficient than $SelectWin$ due to the elimination operation (Step 11) that accelerates the evolution by removing the datasets that are not preferred by GEP individuals.

6. Experimental Study

In this section we assess the performance of our techniques for effective multi-segment sliding windows mining and PIRP. Our algorithms were implemented in Java. All experiments were conducted on an Intel i3 2.20 GHz CPU with 4 GB memory running Windows 7 SP 3.

Datasets We apply our proposed methods to 5 real-world time series datasets containing monthly incidence rates of bacillary dysentery. Due to the sensitivity of the data, we omit the details of the sources and denote the five time series datasets as *China-A*, *China-B*, *China-C*, *China-D* and *China-E*. Each dataset records the monthly incidence rates of bacillary dysentery from January 2004 to December 2010 in a province of China. So, there are 84 time points in total in each dataset.

Moreover, we select two health related datasets, namely *Measlnyc* and *Mumps*, from Time Series Data Library at http://robjhyndman.com/TSDL. The two datasets record monthly reported numbers of cases of measles and mumps in New York City over 40 years, respectively. Firstly, we select 10 years' data from January 1961 to December 1970 to test the effectiveness of the proposed algorithms. The two datasets are denoted as $Measlnyc_{10}$ and $Mumps_{10}$. Later, we will test the scalability of the proposed algorithms by involving longer time interval. As the population of New York City in 1960s

Algorithm 3 VoteWin($DS, pSize, ng_{max}$)

Input: (1) DS: the set of datasets generated by all candidate effective multi-segment sliding windows; (2) $pSize$: the number of individuals in a population; (3) ng_{max}: the maximum number of generations for GEP evolution;

Output:

(η^*, id^*), η^*: an individual (prediction model) with the largest fitness; D^*: the dataset that η^* most prefer.

$ng \leftarrow 1$; // ng indicates the number of evolved generations;
$eg \leftarrow \lfloor \frac{ng_{max}}{|DS|} \rfloor$;
$pop \leftarrow CreateSeedPop(pSize)$; // initialize the population;
$pOrders \leftarrow FitEvaluate(pop, DS)$; // get the p-order of each individual;
$vScores \leftarrow Vote(DS, pOrders)$; // compute the voting score of each dataset in DS;
$\eta^* \leftarrow GetBest(pop)$; // get the individual with the largest fitness in pop;
$D^* \leftarrow RecordDataset(DS)$; // record the dataset that η^* most prefer;
$ng \leftarrow ng + 1$;
while $ng < ng_{max}$ **do**
  if ng mod $eg = 0$ and $|DS| > 1$ **then**
    $DatesetElimination(DS, vScores)$; // eliminate the dataset with the lowest voting score in $DS \backslash \{D^*\}$;
    $VScoreReset(vScores)$; // reset the voting scores;
  end if
  $pop \leftarrow Select(pop)$; // select individuals to compose a new population;
  $GeneticModify(pop)$; // genetic modifications;
  $pOrders \leftarrow FitEvaluate(pop, DS)$;
  $vScores \leftarrow Vote(DS, pOrder)$;
  $\eta \leftarrow GetBest(pop)$;
  $D \leftarrow RecordDataset(DS)$;
  if $fit(\eta, D) > fit(\eta^*, D^*)$ **then**
    $\eta^* \leftarrow \eta$;
    $D^* \leftarrow D$;
  end if
  $ng \leftarrow ng + 1$;
end while

is over 16 million, which is far more than the monthly cases of measles or mumps, we predict the number of cases directly in this experimental study.

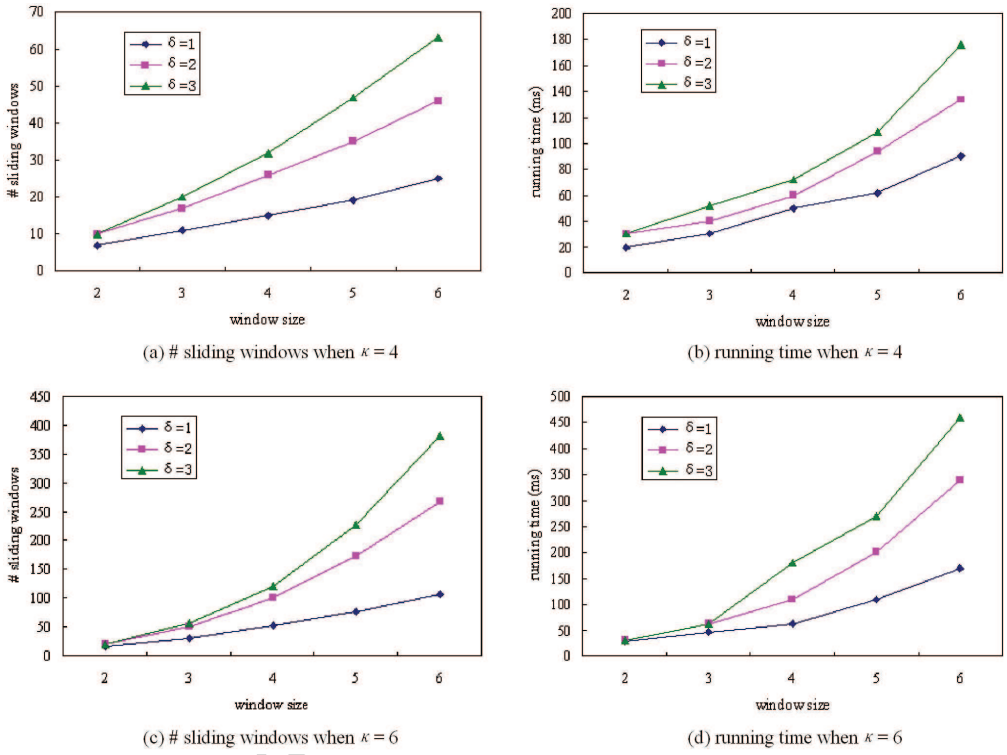
6.1. Effective Multi-segment Sliding Window Discovery

Efficiency test on $EnumWin$ We now evaluate our heuristic algorithm $EnumWin$ for enumerating candidate effective multi-segment sliding windows. The number of candidate effective multi-segment sliding windows depends on three factors: the window size (ℓ), the limit on segment s_{i-1} larger than segment s_i (δ) and segmentation length (κ). Figure 6 illustrates the number of candidate effective multi-segment sliding windows enumerated by $EnumWin$, as well as the running time under different values of ℓ, δ and κ. From Figure 6 (a) and (c), we can see that the number of candidate effective multi-segment sliding windows increases as the values of ℓ, δ and κ increase, and the number of enumerated multi-segment sliding windows increases in a nearly linear manner when δ and κ are fixed. From Figure 6 (b) and (d), we can see that $EnumWin$ can enumerate the candidate effective multi-segment sliding windows efficiently.

Effective multi-segment sliding window mining A total of 32 candidate effective multi-segment sliding windows were enumerated by $EnumWin$ when $\ell = 4$, $\delta = 3$ and $\kappa = 4$. We use a 4-tuple $< |s_{-3}|, |s_{-2}|, |s_{-1}|, |s_0| >$ ($\sum_{i=-3}^{0} |s_i| = \ell$) to record the size of each segment in a sliding window, and represent a multi-segment sliding window. In each dataset generated by a candidate effective multi-segment sliding window, we reserve the last 5 samples as test set, and other samples as training set. The population size in $SelectWin$ is 100. The arithmetic operators involved in GEP evolution include: $+, -, *, /, \sqrt{}$. $SelectWin$ stops evolution when the number of generations is 1000. We run $SelectWin$ 20 times independently on each training set. Table 1 lists the average relative errors of the best evolved models. The minimum training error in a dataset is in bold.

From Table 1, we can see that the effective multi-segment sliding windows are not fixed. The training accuracies are associated with the sliding windows. Thus, mining effective multi-segment sliding window is necessary for improving the prediction precision. Note that, multiple effective multi-segment sliding windows are discovered for $China\text{-}B$ and $China\text{-}C$. (In either $China\text{-}B$ or $China\text{-}C$, the best prediction models evolved over these sliding windows are identical.)

Table 2 lists the best prediction model evolved by $SelectWin$ on each dataset over the effective multi-segment sliding window.



(a) # sliding windows when $\kappa = 4$

(b) running time when $\kappa = 4$

(c) # sliding windows when $\kappa = 6$

(d) running time when $\kappa = 6$

Figure 6: Performance results on enumerating candidate effective multi-segment sliding windows

6.2. Effectiveness of VoteWin

We apply $VoteWin$ to the training sets. The algorithm using Borda Count is denoted by $VoteWin$-B and the algorithm using Copeland's Method is denoted by $VoteWin$-C. The population size is 100. The evolution stops when the number of generations is 1000. We also run $VoteWin$-B and $VoteWin$-C 20 times independently on each training set.

The prediction models discovered by $VoteWin$-B and $VoteWin$-C are the same as the models discovered by $SelectWin$ (see Table 2). Figure 7 illustrates the average running time of $SelectWin$, $VoteWin$-B and $VoteWin$-C for discovering the optimal sliding window and the prediction model on each training set. From Figure 7, we can see that the running time of $VoteWin$-B and $VoteWin$-C are almost equal, and both of them use less than $SelectWin$. The reason is that the optimal sliding window and prediction model cannot

Table 1: Average relative errors in training sets

window	*China-A*	*China-B*	*China-C*	*China-D*	*China-E*	$Measlnyc_{10}$	$Mumps_{10}$
$<3,1,0,0>$	0.2948	0.1509	0.1616	0.1478	0.2293	0.6645	0.4029
$<2,2,0,0>$	0.3039	0.1558	0.1617	0.1465	0.2324	0.6561	0.3990
$<1,3,0,0>$	0.2950	0.1538	0.1595	0.1479	0.2372	0.6582	0.3986
$<3,0,1,0>$	0.2460	0.1286	0.1404	0.1289	0.2425	0.6267	0.4415
$<2,1,1,0>$	0.2499	0.1265	0.1288	0.1215	0.2428	0.6312	0.3822
$<1,2,1,0>$	0.2521	0.1284	**0.1269**	0.1225	0.2372	0.6177	0.3986
$<0,3,1,0>$	0.2523	0.1587	0.1547	0.1775	0.2737	0.6408	0.4189
$<2,0,2,0>$	0.2499	0.1349	0.1408	0.1282	0.2428	0.6303	0.4365
$<1,1,2,0>$	0.2521	0.1284	**0.1269**	0.1225	0.2656	0.6458	0.3986
$<0,2,2,0>$	0.2318	0.1590	0.1552	0.1797	0.2734	0.6400	0.4061
$<1,0,3,0>$	0.2502	0.1426	0.1434	0.1331	0.2656	0.6456	0.4461
$<0,1,3,0>$	0.2291	0.1578	0.1554	0.1819	0.2754	0.6635	0.4261
$<3,0,0,1>$	0.1945	0.1652	0.1411	**0.1042**	0.2054	0.4725	0.2325
$<2,1,0,1>$	0.2094	0.1368	0.1414	0.1207	0.2137	0.4778	0.2414
$<1,2,0,1>$	0.1641	0.1358	**0.1269**	0.1161	0.1731	0.4581	0.2157
$<0,3,0,1>$	0.1721	0.1308	0.1412	0.1204	0.1765	0.4344	0.2119
$<2,0,1,1>$	0.2094	0.1281	0.1338	0.1207	0.2137	0.4778	0.2414
$<1,1,1,1>$	0.2264	**0.1262**	**0.1269**	0.1225	0.2372	0.4808	0.2615
$<0,2,1,1>$	0.1851	0.1304	0.1470	0.1257	0.1813	0.4303	0.2104
$<1,0,2,1>$	0.1434	**0.1262**	0.1321	0.1110	**0.1468**	0.4824	0.1964
$<0,1,2,1>$	**0.1394**	0.1315	0.1440	0.1136	0.1498	0.4992	**0.1908**
$<0,0,3,1>$	0.1426	0.1313	0.1519	0.1140	0.1718	0.5007	0.1928
$<2,0,0,2>$	0.2094	0.1429	0.1492	0.1207	0.2137	0.4565	0.2414
$<1,1,0,2>$	0.2526	0.1358	0.1409	0.1479	0.2372	0.4570	0.2573
$<0,2,0,2>$	0.1851	0.1304	0.1470	0.1257	0.1813	0.4303	0.2104
$<1,0,1,2>$	0.2255	**0.1262**	0.1321	0.1331	0.2429	0.4570	0.2573
$<0,1,1,2>$	0.2148	0.1315	0.1554	0.1720	0.2391	0.4373	0.2623
$<0,0,2,2>$	0.1543	0.1324	0.1640	0.1209	0.1745	**0.4295**	0.1923
$<1,0,0,3>$	0.2572	0.1323	0.1338	0.1661	0.2491	0.4570	0.2557
$<0,1,0,3>$	0.2678	0.1472	0.1378	0.1745	0.2711	0.4373	0.2610
$<0,0,1,3>$	0.2352	0.1508	0.1479	0.1751	0.2398	0.4317	0.2689
$<0,0,0,4>$	0.3274	0.1854	0.1651	0.1720	0.3021	0.4320	0.2740

be found until *SelectWin* finishes the evolution on all datasets. In contrast, *SelectWin* removes non-preferred datasets in the process of prediction mode

Table 2: Prediction model evolved on each training set

time series dataset	*prediction model*
China-A	$R(t) = R(t-12)/(R(t-13)/R(t-1))$, $R(t) \in$ *China-A*, $t \geq 25$
China-B	$R(t) = \sqrt{R(t-12) * R(t-1)}$, $R(t) \in$ *China-B*, $t \geq 37$
China-C	$R(t) = \sqrt{R(t-12) * \sqrt{R(t-24)}}$, $R(t) \in$ *China-C*, $t \geq 37$
China-D	$R(t) = R(t-36) * R(t-1)/R(t-37)$, $R(t) \in$ *China-D*, $t \geq 39$
China-E	$R(t) = R(t-1) * R(t-12)/R(t-13)$, $R(t) \in$ *China-E*, $t \geq 37$
$Measlnyc_{10}$	$R(t) = R(t-1) * R(t-1)/R(t-2)$, $R(t) \in Measlnyc_{10}$, $t \geq 14$
$Mumps_{10}$	$R(t) = R(t-12)/(R(t-13)/R(t-1))$, $R(t) \in Mumps_{10}$, $t \geq 25$

evolution, so that individual evaluation is accelerated. Figure 8 illustrates the average number of generations when the optimal prediction model is discovered. We can see that the optimal prediction models are got within 400 generations in average, and both $VoteWin$-B and $VoteWin$-C can discover the optimal prediction model in less number of generations than $SelectWin$ in most training sets. We conjecture that in $VoteWin$ each individual is evaluated by several datasets, and assigned the highest fitness for selection, so that the individuals with higher fitness are more likely to be selected for further evolution. As a result, the optimal prediction model may be generated in less number of generations.

As the total number of candidate effective multi-segment sliding windows is 32, either $VoteWin$-B or $VoteWin$-C eliminates a window with the lowest voting score every $\lfloor \frac{1000}{32} \rfloor$ generations until only one is left. As stated before, the sliding window with the second lowest voting score will be eliminated, if the window with the lowest voting score is the most preferred one of the best individual. We call this situation a voting conflict. Table 3 presents the average number of voting conflicts in the process of effective multi-segment sliding window mining. From Tables 1 and 3, we can see that the number of voting conflicts is related to the accuracy of prediction model. The conflict occurs rarely when the accuracy of prediction model is high, and vice versa.

Table 3: Average number of voting conflicts

algorithms	*China-A*	*China-B*	*China-C*	*China-D*	*China-E*	$Measlnyc_{10}$	$Mumps_{10}$
$VoteWin$-B	3.50	0.35	7.90	2.90	1.20	13.90	11.60
$VoteWin$-C	3.80	1.00	6.35	1.55	1.70	15.30	11.70

In each running of $VoteWin$, candidate effective multi-segment sliding

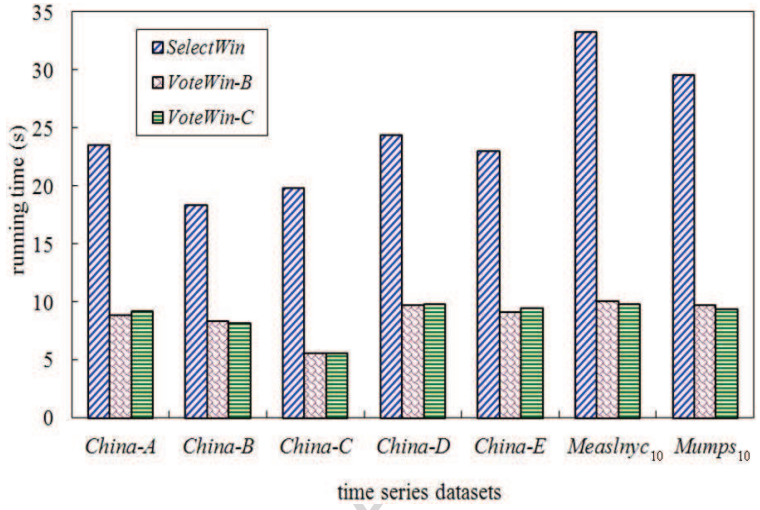


Figure 7: Running time for mining the optimal sliding window and prediction model

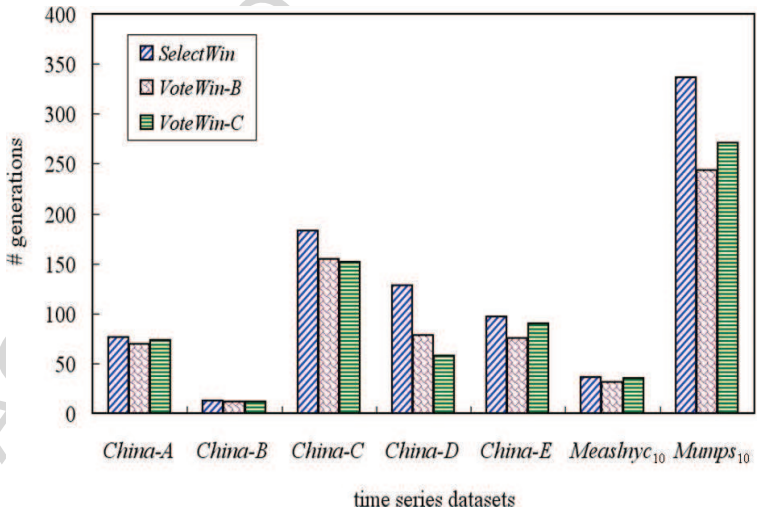


Figure 8: Number of generations for evolving the optimal prediction model

windows are eliminated one by one. We record the elimination order of all candidate effective multi-segment sliding windows in each running of $VoteWin\text{-}B$ and $VoteWin\text{-}C$. The elimination order starts from 1. That

is, the elimination order of the first eliminated sliding window is 1, and the elimination order of the optimal multi-segment sliding window equals to the total number of candidate effective multi-segment sliding windows. Compared with Table 1, we find that for most sliding windows having low training accuracies, the average elimination order is small. In other words, $VoteWin$ eliminates the windows that are not suitable for prediction model evolution as early as possible. Tables A.1 and A.2 in Appendix list the average elimination order of each sliding window eliminated by $VoteWin$-B and $VoteWin$-C, respectively.

6.3. Prediction Accuracy

From the prediction models listed in Table 2, we get the predicted values on the 5 samples in each test set. Moreover, we apply $ARIMA$ and Wavelet-ANN (WNN) for prediction. Table 4 lists the average relative errors between the predicted values and target values in each test set. For each test set, the minimum prediction error is in bold. As the prediction models evolved by $SelectWin$, $VoteWin$-B and $VoteWin$-C are identical, the prediction errors of these three algorithms are the same. The errors in Table 4 show that the prediction models evolved by GEP based algorithm can get higher prediction precisions in all test sets except $China$-C. Thus, it is desirable to apply our proposed algorithms to PIRP problem.

Table 4: Prediction errors on each test set

algorithms	*China-A*	*China-B*	*China-C*	*China-D*	*China-E*	$Measlnyc_{10}$	$Mumps_{10}$
$ARIMA$	0.2421	0.1476	**0.0792**	0.2180	0.3451	3.8319	0.4003
WNN	0.4064	0.1385	0.2509	0.1985	0.5288	1.3585	0.7170
$SelectWin$	**0.1836**	**0.1032**	0.1372	**0.1160**	**0.2439**	**0.2691**	**0.2507**
$VoteWin$-B	**0.1836**	**0.1032**	0.1372	**0.1160**	**0.2439**	**0.2691**	**0.2507**
$VoteWin$-C	**0.1836**	**0.1032**	0.1372	**0.1160**	**0.2439**	**0.2691**	**0.2507**

6.4. Scalability Test

To test the scalability of $SelectWin$ and $VoteWin$, we generate 6 time series datasets covering longer time intervals in $Measlnyc$ and $Mumps$. Specifically, $Measlnyc_{20}$ and $Mumps_{20}$ include the data from January 1951 to December 1970 (20 years), $Measlnyc_{30}$ and $Mumps_{30}$ include the data from January 1941 to December 1970 (30 years), datasets $Measlnyc_{40}$ and $Mumps_{40}$ include the data from January 1931 to December 1970 (40 years).

We use the same experiment settings as the one used in the effectiveness tests on $Measlnyc_{10}$ and $Mumps_{10}$, and apply $SelectWin$ and $VoteWin$ to other datasets containing more data in longer time intervals. Figure 9 illustrates the average running time of $SelectWin$, $VoteWin$-B and $VoteWin$-C for evolving effective multi-segment sliding window on each training set. The running time of $SelectWin$, $VoteWin$-B and $VoteWin$-C increase linearly as more data (in longer time interval) are included for training. So that it is practicable to apply our proposed algorithms to larger datasets.

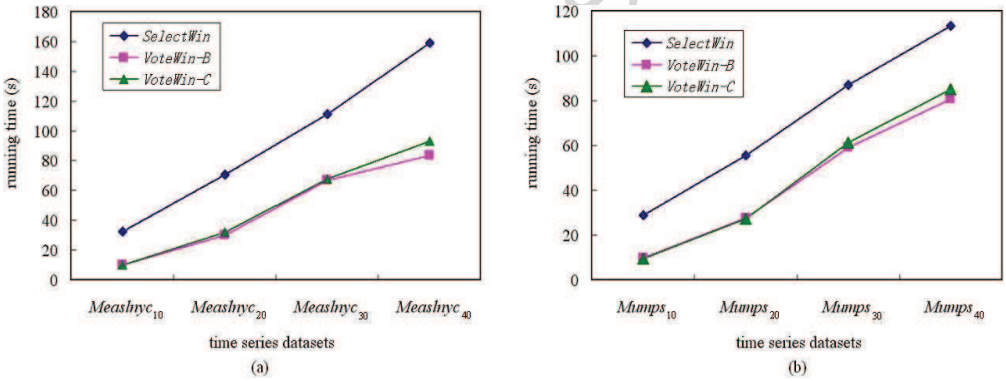


Figure 9: Comparison on the running time for discovering optimal sliding window and the prediction model on different training sets

The optimal sliding windows discovered for $Measlnyc_{20}$, $Measlnyc_{30}$ and $Measlnyc_{40}$ are $< 0, 2, 1, 1 >$, $< 1, 2, 0, 1 >$, $< 0, 2, 1, 1 >$, respectively. The best prediction models evolved over these sliding windows are identical: $R(t) = R(t-1) * R(t-24)/R(t-25)$. The prediction error of this prediction model on the test set is 0.3077 in average, which is worse than the prediction model discovered for $Measlnyc_{10}$ (see Table 4). For $Mumps_{20}$, $Mumps_{30}$ and $Mumps_{40}$, the optimal sliding windows are $< 0, 0, 2, 2 >$, $< 0, 0, 2, 2 >$, $< 1, 0, 2, 1 >$, respectively, and the best prediction models evolved over these sliding windows are identical to the one discovered for $Mumps_{10}$.

From the test results, we can see that the suitable time interval for training the prediction model should be closer to the prediction target and not too long. As the prediction model evolved by GEP may not involve all data in the sliding window, the optimal prediction model for different sliding windows may be identical.

7. Related Works

Time Series Modeling Time series study is distinct from other data mining problems due to the existence of natural temporal ordering in time series data. Time series study is important since scientists can extract meaningful characteristics of the data, and develop the model to predict future data points based on the historical data. Time series study has been widely applied in many domains, such as econometrics [16, 17], medical data analysis [18, 19, 20], meteorology [8]. Representative research topics on time series include: semantics [21], fingerprinting [22], subsequence search [23, 24, 25], anomaly detection [26, 27], similarity measure [28, 29], etc. Research on time series is often part of research on streaming data [30] or on temporal databases [31].

There are a wide range of time series modeling methods in the literature, making it impossible to give a comprehensive overview in this paper. In general, time series modeling methods can be classified into three types: linear model, such as ARMA, ARIMA [32], non-linear, such as ARCH, GARCH [33, 34], and model-free, such as some wavelet transform based methods [35].

Time series modeling for health informatics Traditional time series modeling methods have been widely applied to infectious disease prevention and control [1, 2, 3, 4, 5, 6]. For instance, using time series methods, the authors in [3] develop some models of emergency department utilization for identifying abnormally high visit rates that may be an early warning of a bioterrorist attack. The authors in [1] use ARIMA to predict the number of beds occupied during a SARS outbreak in a tertiary hospital in Singapore. In [2], ARIMA is used to predict the incidence of pulmonary tuberculosis. ANN can overcome the linear-modeling limitation of ARIMA, so it has been applied to many disease incidence predictions, such as cancer and hepatitis [4, 5]. Moreover, reference [6] proposes a hybrid methodology that combines both ARIMA and ANN models to take advantage of the unique strength of ARIMA and ANN models in both linear and nonlinear modeling.

Evolutionary computation for time series modeling As time series prediction can be considered as a particular case of a symbolic regression problem [36], evolutionary computation models have been used for chaotic, nonlinear and empirical time series. For example, Genetic Programming (GP) can be used for modeling and forecasting chaotic time series [37, 38, 39], and discriminating between chaotic signals and noise [40]. GP based time series prediction has been successfully used in a wide range of areas,

such as financial time series [41, 42], traffic data [43], meteorological data [44]. Furthermore, there are some efforts to improve the effectiveness and adaptiveness of GP based time series modeling [45, 46, 47].

GEP has been used successfully to solve various time series problems so far. Besides Ferreira's work in [7], the authors in [12] design a GEP-based method, called Differential by Microscope Interpolation, for sunspot series prediction. In [9], the authors apply an adaptive GEP-based method to predict the precipitation and temperatures in a region of Romania. The authors in [8, 10] perform a comparison between GEP and ARIMA in precipitation modeling and wind prediction, respectively. The experimental studies demonstrate that the results of GEP are satisfactory and better than ARIMA. The authors in [11] develop a GEP system EGIPSYS for symbolic regression problems and demonstrated its utility for time series modeling.

8. Discussions and Conclusions

In this paper, we have introduced the problem of mining effective sliding window, for discovering optimal sliding windows for building accurate prediction model, for GEP based time series modeling. We investigated how to efficiently mine effective multi-segment sliding window, which consists of several segments from different periodical intervals. The main contributions of this paper include designing a heuristic method for enumerating the candidate effective multi-segment sliding windows, proposing GEP based methods to find the optimal sliding window and then produce a mathematical model based on that window. Experiment results show that the proposed methods are efficient and effective. We are not aware of other work on mining such multi-segment sliding window for GEP based time series modeling considered.

To keep our discussion simple, in this paper we only considered using basic arithmetic operators in developing prediction models. More operators can be used in the GEP based model evolution. For example, the authors in [12] applied the differential operator for building the prediction model, and got desirable prediction results on sunspot series. We believe that more accurate prediction models can be evolved by introducing more complex operators.

There are many interesting issues that deserve research effort in the future. For example, it is interesting to consider how to add the environment factors in effective multi-segment sliding window mining, how to describe the

relationships among historical data, and how to evaluate the candidate effective multi-segment sliding windows more efficiently. Moreover, as periodicity exists in many time series data, it is of interest to generalize the proposed methods to solve applications in a more general scenario, including in other domains such as economics, meteorology and finance.

9. Acknowledgement

The authors would like to thank the Editor and anonymous reviewers for their valuable comments to improve this paper, and thank Zijian Feng at Chinese Center for Disease Control and Prevention for his helpful comments. The work described in this paper was partially supported by Natural Science Foundation of China (Grant No. 61103042), Research Fund for the Doctoral Program of Higher Education of China (Grant No. 20100181120029), National Special Foundation for Health Research of China (Grant No. 200802133), and State Key Laboratory of Software Engineering of China (Grant No. SKLSE2012-09-32). Work by Guozhu Dong was supported in part by NSF IIS-1044634.

References

[1] A. Earnest, M. I. Chen, D. Ng, L. Y. Sin, Using autoregressive integrated moving average (ARIMA) models to predict and monitor the number of beds occupied during a SARS outbreak in a tertiary hospital in Singapore, BMC Health Services Research 5 (1) (2005) 36.

[2] L. Meng, Y. Wang, Application of ARIMA model on prediction of pulmonary tuberculosis incidence, Chinese Journal of Health Statistics 27 (5) (2010) 507–509.

[3] B. Y. Reis, K. D. Mandl, Time series modeling for syndromic surveillance, BMC Medical Informatics and Decision Making 3 (1) (2003) 2.

[4] P. Guan, D.-S. Huang, B.-S. Zhou, Forecasting model for the incidence of hepatitis a based on artificial neural network, World Journal of Gastroenterology 10 (24) (2004) 3579–3582.

[5] J. Khan, J. S. Wei, M. Ringnér, L. H. Saal, M. Ladanyi, F. Westermann, F. Berthold, M. Schwab, C. R. Antonescu, C. Peterson, P. S. Meltzer,

Classification and diagnostic prediction of cancers using gene expression profiling and artificial neural networks, Nature Medicine 7 (2001) 673–679.

[6] G. Zhang, Time series forecasting using a hybrid ARIMA and neural network model, Neurocomputing 50 (2003) 159–175.

[7] C. Ferreira, Gene expression programming: A new adaptive algorithm for solving problems, Complex Systems 13 (2) (2001) 87–129.

[8] A. Barbulescu, E. Bautu, ARIMA models versus gene expression programming in precipitation modeling, in: Proceedings of the 10th WSEAS Internationl Conference on Evolutionary Computing, Prague, Czech Republic, 2009, pp. 112–117.

[9] A. Barbulescu, E. Bautu, Time series modeling using an adaptive gene expression programming algorithm, International Journal of Mathematical Models and Methods in Applied Sciences 3 (2) (2009) 85–93.

[10] J. J. Flores, M. Graff, E. Cadenas, Wind prediction using genetic programming and gene expression programming, in: Proceedings of the International Conference on Modelling and Simulation in the Enterprises (AMSE 2005), Morelia, Mexico, 2005.

[11] H. S. Lopez, W. R. Weinert, A gene expression programming system for time series modeling, in: Proceedings of XXV Iberian Latin American Congress on Computational Methods in Engineering, Recife, Brazil, 2004.

[12] J. Zuo, C. Tang, C. Li, C. Yuan, A. Chen, Time series prediction based on gene expression programming, in: Proceedings of the 5th International Conference on Web-Age Information Management (WAIM 2004), Dalian, China, 2004, pp. 55–64.

[13] R. Farquharson, Theory of Voting, Yale University Press, Blackwell, 1969.

[14] P. D. Straffin, Jr., Topics in the Theory of Voting, Birkhäuser, Boston, MA, 1980.

[15] K. J. Arrow, Social Choice and Individual Values, 2nd Edition, Yale University Press, 1963.

[16] D. H. Dorr, A. M. Denton, Establishing relationships among patterns in stock market data, Data and Knowledge Engineering 68 (3) (2009) 318–337.

[17] H. J. Teoh, C.-H. Cheng, H.-H. Chu, J.-S. Chen, Fuzzy time series model based on probabilistic approach and rough set rule induction for empirical research in stock markets, Data and Knowledge Engineering 67 (1) (2008) 103–117.

[18] F. Alonso, J. P. Caraça-Valente, L. Martínez, C. Montes, Discovering similar patterns for characterising time series in a medical domain, in: Proceedings of the 1st IEEE International Conference on Data Mining (ICDM 2001), San Jose, CA, USA, 2001.

[19] S. Hirano, S. Tsumoto, Mining similar temporal patterns in long time-series data and its application to medicine, in: Proceedings of the 2nd IEEE International Conference on Data Mining (ICDM 2002), Maebashi City, Japan, 2002.

[20] S. Hirano, S. Tsumoto, Cluster analysis of time-series medical data based on the trajectory representation and multiscale comparison techniques, in: Proceedings of the 6th IEEE International Conference on Data Mining (ICDM 2006), Hong Kong, China, 2006.

[21] P. Wang, H. Wang, W. Wang, Finding semantics in time series, in: Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD 2011), Athens, Greece, 2011, pp. 385–396.

[22] L. Li, B. A. Prakash, C. Faloutsos, Parsimonious linear fingerprinting for time series, Proceedings of the VLDB Endowment 3 (1) (2010) 385–396.

[23] K. Bhaduri, Q. Zhu, N. C. Oza, A. N. Srivastava, Fast and flexible multivariate time series subsequence search, in: Proceedings of the 10th IEEE International Conference on Data Mining (ICDM 2010), Sydney, Australia, IEEE Computer Society, 2010, pp. 48–57.

[24] W.-S. Han, J. Lee, Y.-S. Moon, H. Jiang, Ranked subsequence matching in time-series databases, in: Proceedings of the 33rd International Conference on Very Large Data Bases (VLDB 2007), University of Vienna, Austria, 2007, pp. 423–434.

[25] H. Wu, B. Salzberg, G. C. Sharp, S. B. Jiang, H. Shirato, D. R. Kaeli, Subsequence matching on structured time series data, in: Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD 2005), Baltimore, Maryland, USA, 2005, pp. 682–693.

[26] P. K. Chan, M. V. Mahoney, Modeling multiple time series for anomaly detection, in: Proceedings of the 5th IEEE International Conference on Data Mining (ICDM 2005), Houston, Texas, USA, IEEE Computer Society, 2005, pp. 90–97.

[27] V. Guralnik, J. Srivastava, Event detection from time series data, in: Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD 1999), San Diego, CA, USA, 1999, pp. 33–42.

[28] J. P. Caraça-Valente, I. Lopez-Chavarrias, Discovering similar patterns in time series, in: Proceedings of the sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD 2000), Boston, MA, USA, 2000, pp. 497–505.

[29] D. Gunopulos, G. Das, Time series similarity measures and time series indexing, in: Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD 2001), Santa Barbara, CA, USA, 2001, p. 624.

[30] M. Kontaki, A. N. Papadopoulos, Y. Manolopoulos, Adaptive similarity search in streaming time series with sliding windows, Data and Knowledge Engineering 63 (2) (2007) 478–502.

[31] J. Y. Lee, R. Elmasri, J. Won, An integrated temporal data model incorporating time series concept, Data and Knowledge Engineering 24 (3) (1998) 257–276.

[32] P. Brockwell, R. Davies, Introduction to Time Series, Springer, New York, 2002.

[33] T. Bollerslev, Generalized autoregressive conditional heteroskedasticity, Journal of Econometrics 31 (1986) 307–327.

[34] R. S. Hacker, A. Hatemi-J, A test for multivariate ARCH effects, Applied Economics Letters 12 (7) (2005) 411–417.

[35] C. K. Chui, An Introduction to Wavelets, Academic Press, San Diego, 1992.

[36] Y. Chen, G. Dong, J. Han, B. W. Wah, J. Wang, Multi-dimensional regression analysis of time-series data streams, in: Proceedings of 28th International Conference on Very Large Data Bases (VLDB 2002), Hong Kong, China, 2002, pp. 323–334.

[37] G. Lee, Time series perturbation by genetic programming, in: Proceedings of the 2001 Congress on Evolutionary Computation, Seoul, Korea, 2001, pp. 403–409.

[38] H. Oakley, Two scientific applications of genetic programming: Stack filters and non-linear equation fitting to chaotic data, in: K. E. Kinnear, Jr. (Ed.), Advances in genetic programming, MIT Press, Cambridge, MA, USA, 1994, Ch. 17, pp. 369–389.

[39] G. G. Szpiro, Forecasting chaotic time series with genetic algorithms, Physical Review E 55 (3) (1997) 2557–2568.

[40] D. B. Fogel, L. J. Fogel, Preliminary experiments on discriminating between chaotic signals and noise using evolutionary programming, in: Proceedings of the 1st Annual Conference on Genetic Programming, Stanford, CA, USA, 1996, pp. 512–520.

[41] M. Kaboudan, A measure of time series predictability using genetic programming applied to stock returns, Journal of Forecasting 18 (1999) 345–357.

[42] M. Santini, A. Tettamanzi, Genetic programming for financial time series prediction, in: Proceedings of the 4th European Conference on Genetic Programming (EuroGP 2001), Lake Como, Italy, 2001, pp. 361–370.

[43] D. Howard, S. C. Roberts, Application of genetic programming to motorway traffic modelling, in: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2002), New York, USA, 2002, pp. 1097–1104.

[44] K. Rodriguez-Vazques, Genetic programming in time series modeling: an application to meteorological data, in: Proceedings of the 2001 Congress on Evolutionary Computation, Seoul, Korea, 2001, pp. 261–266.

[45] S. Eklund, Time series forecasting using massively parallel genetic programming, in: Proceedings of the 17th International Parallel and Distributed Processing Symposium (IPDPS 2003), Nice, France, 2003, pp. 22–26.

[46] D. Rivero, J. R. Rabunal, J. Dorado, A. Pazos, Time series forecast with anticipation using genetic programming, in: Proceedings of the 8th International Work-Conference on Artificial Neural Networks (IWANN 2005), Vilanova i la Geltrú, Barcelona, Spain, 2005, pp. 968–975.

[47] I. Yoshihara, T. Aoyama, M. Yasunaga, GP-based modeling method for time series prediction with parameter optimization and node alternation, in: Proceedings of the 2000 Congress on Evolutionary Computation, La Jolla, CA, USA, 2000, pp. 1475–1481.

Appendix A.

Table A.1: Average elimination order of sliding windows eliminated by $VoteWin\text{-}B$

window	*China-A*	*China-B*	*China-C*	*China-D*	*China-E*	$Measlnyc_{10}$	$Mumps_{10}$
$<3,1,0,0>$	3.30	16.90	22.70	9.20	1.10	9.60	8.20
$<2,2,0,0>$	4.60	20.30	21.70	4.90	2.50	11.00	5.60
$<1,3,0,0>$	5.00	7.40	20.70	6.80	2.40	12.00	9.10
$<3,0,1,0>$	15.70	20.60	31.00	13.20	4.80	1.00	7.10
$<2,1,1,0>$	23.70	29.90	30.00	11.80	12.50	3.00	5.30
$<1,2,1,0>$	22.70	26.20	29.30	20.70	9.90	5.80	3.70
$<0,3,1,0>$	22.30	12.80	25.80	21.20	11.10	8.50	10.20
$<2,0,2,0>$	15.20	24.30	29.00	5.90	11.00	2.00	3.30
$<1,1,2,0>$	20.30	27.80	29.10	15.40	13.60	5.40	1.80
$<0,2,2,0>$	19.20	15.70	23.90	17.50	14.50	8.20	10.40
$<1,0,3,0>$	22.50	14.50	27.40	15.80	7.90	4.30	1.30
$<0,1,3,0>$	22.10	10.40	24.70	19.00	12.80	7.20	12.20
$<3,0,0,1>$	20.20	7.40	16.10	32.00	24.40	16.70	22.20
$<2,1,0,1>$	18.40	23.60	19.30	22.90	26.20	16.60	24.70
$<1,2,0,1>$	24.90	18.70	16.30	28.10	28.70	24.10	27.30
$<0,3,0,1>$	19.00	7.30	7.70	22.80	22.50	19.10	17.20
$<2,0,1,1>$	20.90	27.20	18.60	19.30	22.40	17.20	23.50
$<1,1,1,1>$	25.50	30.40	18.50	26.80	27.80	23.00	28.20
$<0,2,1,1>$	19.90	22.50	11.30	22.60	21.20	26.30	18.10
$<1,0,2,1>$	27.30	29.30	13.20	30.40	32.00	24.00	28.40
$<0,1,2,1>$	32.00	19.10	8.10	28.50	27.50	28.90	32.00
$<0,0,3,1>$	20.80	10.50	4.70	25.30	25.30	13.30	14.20
$<2,0,0,2>$	8.30	12.80	18.50	9.40	18.70	20.10	23.60
$<1,1,0,2>$	11.80	20.50	12.40	13.50	19.70	23.60	29.90
$<0,2,0,2>$	4.60	8.80	8.40	2.20	10.10	28.10	18.70
$<1,0,1,2>$	13.00	25.70	13.30	12.30	16.10	22.00	28.70
$<0,1,1,2>$	7.70	17.10	6.00	7.70	10.00	30.00	20.90
$<0,0,2,2>$	2.30	10.30	1.70	4.90	6.80	32.00	15.30
$<1,0,0,3>$	21.10	4.00	10.00	21.50	26.20	22.40	28.50
$<0,1,0,3>$	15.20	2.90	4.30	6.90	22.40	30.90	20.10
$<0,0,1,3>$	11.70	2.10	1.50	17.70	18.30	14.90	15.50
$<0,0,0,4>$	6.80	1.00	2.80	11.80	15.60	17.10	13.00

Table A.2: Average elimination order of sliding windows eliminated by $VoteWin\text{-}C$

window	*China-A*	*China-B*	*China-C*	*China-D*	*China-E*	$Measlnyc_{10}$	$Mumps_{10}$
$<3,1,0,0>$	2.70	21.30	22.50	8.10	1.10	10.00	8.00
$<2,2,0,0>$	5.60	19.80	21.40	4.40	2.60	11.00	5.50
$<1,3,0,0>$	5.30	8.10	20.30	6.70	2.30	12.00	9.00
$<3,0,1,0>$	12.90	23.50	31.00	12.50	5.10	1.00	7.00
$<2,1,1,0>$	21.70	29.90	30.00	13.70	13.20	3.00	5.30
$<1,2,1,0>$	20.30	27.60	29.10	20.90	10.60	5.90	3.60
$<0,3,1,0>$	16.60	15.70	25.60	20.10	11.60	8.50	10.20
$<2,0,2,0>$	12.00	22.90	29.00	7.40	11.70	2.00	3.40
$<1,1,2,0>$	16.90	25.80	28.10	17.40	14.10	5.20	1.80
$<0,2,2,0>$	14.70	15.40	23.60	16.90	14.20	8.10	10.80
$<1,0,3,0>$	18.70	18.50	27.40	17.20	8.10	4.20	1.40
$<0,1,3,0>$	17.40	12.60	24.60	18.90	13.20	7.10	12.00
$<3,0,0,1>$	24.80	7.30	16.00	32.00	24.80	15.70	22.10
$<2,1,0,1>$	19.30	22.50	18.90	21.10	26.90	16.60	24.80
$<1,2,0,1>$	27.90	21.60	15.90	28.70	28.40	23.70	27.30
$<0,3,0,1>$	24.40	9.10	7.50	21.10	22.90	19.60	17.10
$<2,0,1,1>$	22.90	26.70	19.10	18.50	25.60	18.00	24.00
$<1,1,1,1>$	27.50	31.50	21.90	26.30	27.80	23.80	27.90
$<0,2,1,1>$	24.70	19.40	11.50	21.10	21.00	27.00	18.10
$<1,0,2,1>$	29.00	30.40	13.60	31.00	32.00	25.00	28.60
$<0,1,2,1>$	32.00	21.30	8.20	27.50	27.10	29.00	32.00
$<0,0,3,1>$	27.20	12.40	5.20	24.40	24.50	13.00	14.30
$<2,0,0,2>$	6.50	10.20	18.30	8.50	19.50	19.70	23.10
$<1,1,0,2>$	9.20	17.30	12.60	13.10	19.30	22.80	29.80
$<0,2,0,2>$	3.60	5.80	8.70	2.50	8.70	28.10	18.80
$<1,0,1,2>$	8.70	20.70	12.30	12.20	14.40	22.40	28.80
$<0,1,1,2>$	5.20	12.80	5.70	6.60	9.40	30.00	20.90
$<0,0,2,2>$	2.30	7.20	1.60	3.80	5.60	32.00	15.50
$<1,0,0,3>$	23.30	4.60	9.90	22.00	25.50	22.50	28.60
$<0,1,0,3>$	19.00	2.90	4.10	12.80	22.60	30.90	20.10
$<0,0,1,3>$	14.70	2.20	1.50	16.50	18.20	14.00	15.20
$<0,0,0,4>$	11.00	1.00	2.90	14.10	16.00	16.20	13.00

Biography

Lei Duan received his BS and PhD degrees both in Computer Science from Sichuan University in 2003 and 2008, respectively. He was a visiting PhD student in Department of Computer Science and Engineering at Wright State University from 2007 to 2008. He joined the School of Computer Science at Sichuan University as a faculty member in 2009. He is currently a visiting scholar in School of Computing Science at Simon Fraser University from 2012 to 2013. His research interests include data mining, knowledge management, evolutionary computation, bioinformatics and health-informatics.

Changjie Tang is a professor, the director of the institute of database and knowledge engineering in School of Computer Science at Sichuan University, and is vice director of China Computer Federation Technical Committee on Databases. His research interests include database theory, data mining and knowledge discovery, data cube and OLAP, information security.

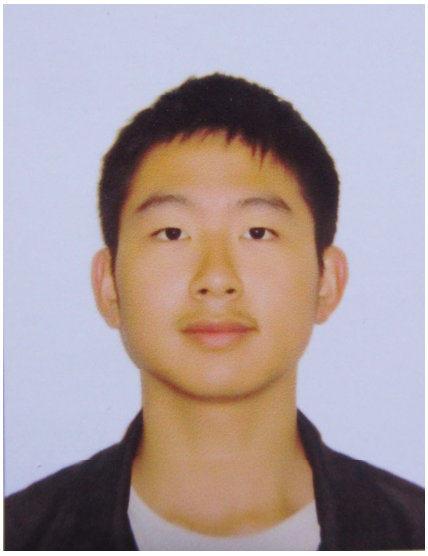
Xiaosong Li is a professor, the dean of West China School of Public Health at Sichuan University, and is the president of the 4th West China Hospital. He is a fellow of the royal statistical society of UK, and is the elected chair of China's deans council of public health school and the editor-in-chief of the journal of contemporary preventive Medicine. His main research interests include statistical and epidemiological methodology, applications of multilevel statistical modeling in health service and system, and health policy and decision making.

Guozhu Dong is a full professor at Wright State University. He earned a PhD in Computer Science from the University of Southern California. His main research interests are data mining, bioinformatics, and databases. He has published over 130 articles and two books entitled "Sequence Data Mining" and "Contrast Data Mining", and he holds 4 US patents. He is widely known for his pioneering work on contrast/emerging pattern mining and applications and for his work on first-order maintenance of recursive and transitive closure views. He is a senior member of both IEEE and ACM.

Xianming Wang is currently a Master student in School of Computer Science at Sichuan University. His research interests include data mining, natural language processing, and high performance computing.

Jie Zuo is an associate professor in School of Computer Science at Sichuan University. He received his PhD degree in Computer Science from Sichuan University in 2005. His research interests include database system, OLAP and data mining, evolutionary computation, and data warehousing.

Min Jiang is currently a PhD student in Department of Epidemiology and biostatistics in West China School of Public Health at Sichuan University. She received her MS degree in Epidemiology and biostatistics in 2008 from Sichuan University. Her research interests include the application of statistical methods in epidemiology and public health, data mining.



Zhongqi Li is currently a junior student in Computer Science and Technology in School of Computer Science at Sichuan University. His research interests include data mining and web information processing.

Yongqing Zhang is currently a PhD student in School of Computer Science at Sichuan University. His research interests include machine learning and bioinformatics.