Proceedings of the 2007 IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB 2007)

Gene Relation Discovery by Mining Similar Subsequences in Time-Series Microarray Data

Vincent S. Tseng, Lien-Chin Chen, Jian-Jie Liu
Department of Computer Science and Information Engineering, National Cheng-Kung University, Taiwan, R.O.C.
{ vincent, cljimmy, kyrill }@idb.csie.ncku.edu.tw

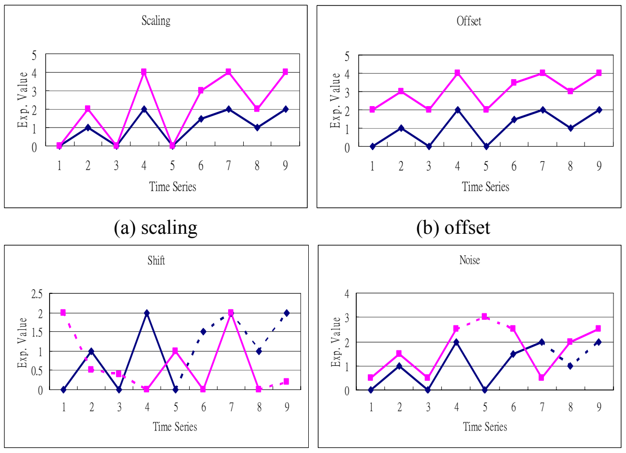
Abstract*—Time-series microarray techniques are newly used to monitor large-scale gene expression profiles for studying biological systems. Previous studies have discovered novel regulatory relations among genes by analyzing time-series microarray data. In this study, we investigate the problem of mining similar subsequences in time-series microarray data so as to discover novel gene relations. A functional relationship among genes often presents itself by locally similar and potentially time-shifted patterns in their expression profiles. Although a number of studies have been done on time-series data analysis, they are insufficient in handling four important issues for time-series microarray data analysis, namely scaling, offset, shift, and noise. We proposed a novel method to address the four issues simultaneously, which consists of three phase, namely angular transformation, symbolic transformation and suffix-tree-based similar subsequences searching. Through experimental evaluation, it is shown that our method can effectively discover biological relations among genes by identifying the similar subsequences. Moreover, the execution efficiency of our method is much better than other approaches.

I. INTRODUCTION

Recently, microarray technologies (particularly oligonucleotide and cDNA arrays) make it possible to monitor the mRNA levels of thousands of gene expressions in a single experiment. The discovery of significant gene pairs with highly-correlated relations may provide valuable insights for biologists in predicting novel biological relations [3]. Moreover, two genes with similar subsequences in time-series microarray data are very likely to be involved in the same regulatory process, as reported in the literature [4, 17, 20, 25].

In time-series microarray analysis, there are four important issues, namely *scaling* (or *amplitude scaling*), *offset*, *shift*, and *noise*, as illustrated in Fig. 1. Fig. 1(a), 1(b) and 1(c) show the phenomenon of amplitude scaling, offset in expression values and shift between two similar subsequences, respectively. Fig. 1(d) illustrates the issue that there could have noisy data points in two similar subsequences.

In recent years, time-series data analysis has been widely applied in various domains like biomedical applications and scientific data analysis. Some interesting research topics, like clustering, classification, similarity search, prediction, and etc. [8, 11, 12, 23], have also been applied to different mining demands on time-series data. Among these research issues, the



(a) scaling (b) offset

(c) shift (d) noise

Fig. 1. Examples of the four problems in time-series analysis.

problem of searching similar subsequences among time sequences is an important one due to wide applications [2, 6, 15, 16, 18, 24].

A number of methods for time-series data analysis have been proposed. In the aspect of sequence transformation, *Symbolic Aggregate approXimation* (SAX) method [13] provides the point of view for transforming numerical data into symbolic representation such that efficient string algorithm can be applied to enhance the performance and scalability. For time sequence matching, some efficient methods were proposed for subsequence matching like *I-Adaptive* [6], *Dual Match* [16], *General Match* [15], the Agrawal's method [1], and *Time-lagged method* [10]. Nevertheless, these methods can not provide a complete solution to address the above-mentioned four problems simultaneously.

In this paper, we propose an efficient and flexible method that can find similar subsequences among time sequences such that it can be applied on time series microarray data for discovering novel gene relations. Moreover, the problems of scaling, offset, shift, and noise are addressed simultaneously by our method. The proposed method consists of three phases, namely i) *angular transformation*, ii) *symbolic transformation,* and iii) *similar subsequence searching*. By angular transformation, the *scaling* and *offset* problems are resolved and also keep the properties of original time series. In the symbolic transformation phase, the *shift* and *noise* problems are solved and can speed up the searching for similar subsequences as well.

Finally, we use suffix tree traversal technique to search the similar subsequences between two time sequences. Through experimental evaluation on real microarray datasets, our method is shown to outperform other existing methods like Agrawal's method [1] and *Time-lagged method* [10] substantially in terms of efficiency. Moreover, the validity of our method in discovering regulatory relations among genes is verified by utilizing the known gene regulatory relations [7].

The reset of this paper is organized as follows. In Section 2, we formally describe the problem definition and the similarity model for time sequences. The proposed method is described in details in Section 3. In Section 4, we present the experimental results on a real biological dataset. Finally, concluding remarks and future work are given in Section 5.

II. Problem Definition

Given two time sequences S and Q which may have different length, the goal is to find all similar subsequence pairs between two time sequences with minimum length threshold, while the factors of scaling, shift, offset, and noise are also considered simultaneously. In the following, we describe some relevant definitions for the targeted problem.

Definition 1 (*Time Sequence*). A time sequence S, S ($s_1s_2s_3 \dots s_{l(S)}$), is an ordered set of real values, where s_i is the i^{th} element of S, and $l(S)$ is sequence length of S.

Definition 2 (*Time Subsequence*). A time subsequence is an ordered sequence. $S_{i,j} = s_is_{i+1}s_{i+2} \dots s_j$, denotes the time subsequence of a time sequence S, which contains the elements of S in positions i through j, and the length of $S_{i,j}$ is $l(S_{i,j}) = i + j - 1$.

Definition 3 (*Similar Time Sequence*). Two time sequences S and Q are called *similar* if and only if $\Phi (S, Q) \geq \Delta$, where $\Phi()$ is a function for calculating similarity between S and Q, and Δ is a specified threshold value.

Definition 4 (*Similar Subsequence*). Given two time sequences S and Q, the sequences S' and Q' are called a similar subsequence of S and Q if and only if S' and Q' are similar and they are subsequences of S and Q, respectively.

Definition 5 (*Distance of Similar Subsequence*). If S and Q have similar subsequences $S_{i,j}$ and $Q_{x,y}$, then the distance between $S_{i,j}$ and $Q_{x,y}$ is $D(S_{i,j} , Q_{x,y}) = | i - x |$.

Notice that any kind of measure could be used for the similarity function $\Phi()$ in Definition 3. In many applications, the most used similarity measures could be *Euclidean Distance* and *Pearson Correlation Coefficient* (PCC). Based on the above definitions, the shape or trend of two subsequences should be very close if they are similar subsequences.

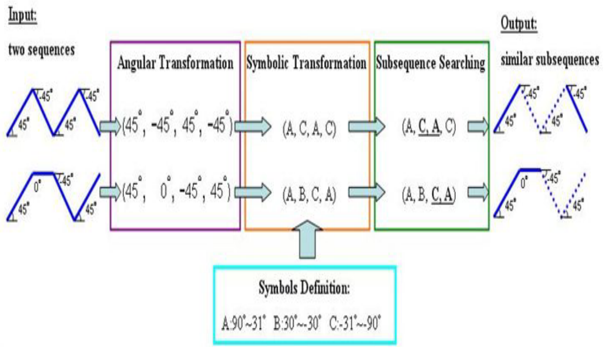


Fig. 2. Workflow of the proposed method.

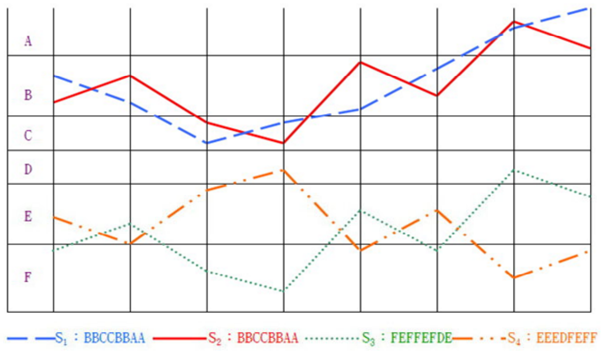


Fig. 3. An example in SAX

III. Proposed Method

The proposed approach for mining similar subsequences between two sequences is decomposed into three sequential phases: i) Angular Transformation, ii) Symbolic Transformation, and iii) Searching Similar Subsequence. Fig. 2 presents the workflow of the proposed method.

Given two time sequences as the initial input, we first apply the angular transformation to convert the original time sequences into angle sequences. Through the angular transformation, the *scaling* and *offset* are resolved, and also can keep the properties of original time series. Second, for the symbolic transformation phase, we divide the range of angles (i.e. -90 to 90 degrees) into several equivalent partitions. Each partition, which represents an angle region, is denoted as an independent symbol. In this way, we convert the angle sequence into the symbol sequence such that the factors of *shift*, and *noise* can be handled effectively. Moreover, this transformation can speed up searching for similar subsequences substantially. Finally, we use a suffix-tree-based approach to search the similar subsequences between two input time sequences.

A. Angular Transformation

Eamonn *et al.* [13] proposed the SAX method for searching similar subsequences efficiently. However, SAX can not distinguish the similar patterns in some cases. Fig. 3 shows an example that illustrates the weakness of SAX method.

By using SAX, S_1 is considered as more similar with S_2 than with *S3*. Consequently, the offset problem could not be solved by SAX. In order to resolve the above problem, we redefine the meaning of similar subsequence as follows:

Definition 6 (*Intermediate Sequence*). For a time sequence $S = s_1s_2s_3...s_{l(S)}$, the *intermediate sequence S'* is

$$S' = s'_1 s'_2 s'_3 ... s'_{l(S)-1}$$

, where
$$s'_i = \frac{s_{i+1} - s_i}{t_{i+1} - t_i}, 1 \le i \le l(S) - 1 \qquad (1)$$

The intermediate sequence is a slope sequence of a pair of two concatenate time points. However, it is possible that all slopes may be in some region. For example, if all time series are in decreasing trend, then the slopes in all intermediate sequences will be of minus values. In order to normalize the slope distribution, which is an important factor for next symbolic transformation phase, we use the *Min-max Normalization* method to transform the intermediate sequence into normalized intermediate sequence after transforming the raw time-series sequence into the intermediate sequence.

Definition 7 (*Normalized Intermediate Sequence*). Given an intermediate sequence $S' = s'_1 s'_2 s'_3 ... s'_{l(S)-1}$ that contains *l(S)*-1 dimensions, and the *Normalized Intermediate Sequence* is the following.

$$s''_i = \frac{s'_i - \min}{\max - \min}(new\_\max - new\_\min) + new\_\min \qquad (2)$$

, where *min* and *max* are the minimum and maximum values of all slopes in all of the intermediate sequences, respectively. The *new_min* and *new_max* represent the minimum and maximum values that we want in all of the normalized intermediate sequences, respectively. In this work, the default values of *new_min* and *new_max* are equal to -1 and 1, respectively.

$$s''_i = 2 \times \frac{s'_i - \min}{\max - \min} - 1 \qquad (3)$$

Finally, we use the inverse function of sine (i.e. $\sin^{-1}$) to transform normalized intermediate sequence into *angle sequence*.

B. Symbolic Transformation

The purpose of symbolic transformation is to convert angle sequence into a string sequence such that efficient string alignment algorithms can be applied to search similar subsequences. In order to distribute the angle values in an angle sequence in normal distribution, we apply *z-score normalization* to all angle values. The z-score associated with the i^{th} observation of a random variable *x* is given by

$$z_i \equiv \frac{x_i - \bar{x}}{\sigma} \qquad (4)$$

, where $\bar{x}$ is the mean and σ is the standard deviation of all observations x_1, ..., x_n. Since the angle sequence is in normal distribution, we can divide the angle region into *SN* partitions and denote each partition with one different symbol.

C. Searching Similar Subsequence by using Suffix Tree

We build the suffix tree [14, 22] to find all (non-overlapped) similar pairs in two sequences with transformed symbol sequence. We illustrate our method through the following example.

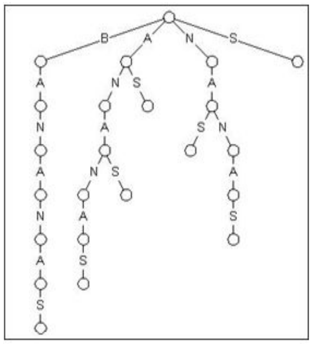
Example 1 Given a transformed symbol sequence, *S*("BANANAS"), which indicates the resulted sequence after angular and symbolic transformations, we want to find similar subsequences in another transformed symbol sequence *T*("BASCANA").

First, we build a suffix tree *sufftree(S)* for sequence *S*("BANANAS") by inserting all subsequences starting from i^{th} position through $l(S)^{th}$ position as shown in Fig. 4(a). Then, we apply tree traversal process on each subsequence of *T*("BASCANA") on *sufftree(S)*. For each suffix sequence of *T* (i.e. *T'*), we start from 1st position and linearly search common subsequence Seq_{com} in *sufftree(S)*. When we find a Seq_{comm} at starting position *m* and *n* in *T'* and *S*, respectively, we record a *starting position pair* (SPP) ({*m*}, {*n*}) for Seq_{comm}, which we denote it as $SPP(Seq_{comm})$ = ({*m*}, {*n*}). Next, we start at $(m+ l(Seq_{comm}))^{th}$ position of *T'*, and search common subsequence again. Note that it is possible that a Seq_{comm} of *T'* has more than one subsequences in *S*, and Seq_{comm} may appear multiple times in *T'*. Therefore, each entry of SPP is a collection set which may have multiple start positions.

In our example, the first suffix sequence *T'* of *T*("BASCANA") is the whole sequence "BASCANA". For *T'*("BASCANA") and *S*, the first Seq_{comm} is "BA", when *m*=1 and *n*=1. We record first SPP of Seq_{comm} ("BA") as ({1}, {1}), i.e. *SPP("BA")*= ({*1*}, {*1*}). In next iteration, we start at 3rd position of *T'*, i.e. "S", and record ({3},{7}) for common subsequence "B". The next Seq_{comm} is "C". However, "C" does not appear in *S*. Therefore, we record *SPP("C")*= ({*4*}, {}). The empty set {} represents Seq_{comm} which can not be matched in suffix tree *sufftree(S)*. Next, we start from "A" which is at 5th position of *T'*("BASCANA"). The Seq_{comm} is "ANA" and is matched at 2nd and 4th positions of *S*("BANANAS"). Then, we record *SPP("ANA")*= ({*5*}, {*2, 5*}). While *T'* is searched to end position, we replace *T'* by next subsequence "ASCANA" of *T*("BASCANA"), and search all SPP for new *T'* and *S* in above way. By repeating the above process for each suffix sequence of *T*("BASCANA"), we can find all matched non-overlapped subsequences as shown in Fig. 4(b).

Next, we filter the duplicates of common subsequence Seq_{comm} which is associated with common SPP in Fig. 4(c). We also apply coverage filtering that is to keep maximum common subsequence Sub_{max} in length and filter out the Seq_{comm} which is the subsequence of Sub_{max}. For instance, "ANA" is maximum common subsequence for "NA". It is easy to discovery SPP("ANA")=({5}, {2, 4}) since SPP("NA")=({6},{3, 5}) are concatenations of common subsequence in start position index. In other words, "ANA" is the prefix sequence of "NA". Finally, we find the similar subsequence of *S* and *T* as follows:

S=(BA)N(ANA)S



(a)

Seq_{comm}	*SPP*
BA	({1}, {1})
AS	({2}, {6})
S	({3}, {7})
C	{{4}, {}}
ANA	({5}, {2, 4})
NA	({6}, {3, 5})

(c)

T'	Seq_{comm}	*SPP*	Step
BASCANA	BA	({1}, {1})	1
	S	({3}, {7})	2
	C	{{4}, {}}	3
	ANA	({5}, {2, 4})	4
ASCANA	AS	({2}, {6})	5
	C	({4}, {})	6
	ANA	({5}, {2, 4})	7
SCANA	S	({3}, {7})	8
	C	({4}, {})	9
	ANA	({5}, {2, 4})	10
CANA	C	({4}, {})	11
	ANA	({5}, {2, 4})	12
ANA	ANA	({5}, {2, 4})	13
NA	NA	({6}, {3, 5})	14

(b)

Fig. 4. An example for searching similar sub-sequences.

T=(BA)SC(ANA)

That is, we get "BA" and "ANA" as similar subsequences.

D. Algorithm and Complexity Analysis

Fig. 5 presents the algorithm of our approach. The inputs are two time-series sequences and the parameters shown in Table I, and the output returns all similar subsequences between these two time sequences. The four main problems in time series analysis, namely scaling, offset, shift, and noise, are resolved in our method. The *scaling* and *offset* problems can be handled through angular transformation. Since we use the suffix tree structure to search the similar subsequences, it is clear to know that *offset* and *noise* problems are resolved, too.

Given two time sequences *S* and *T* with length *n* and *m*, respectively, the time-complexity of building a suffix-tree for sequence *S* is O(n^2), and the time-complexity for searching similar subsequences is O($n \times m$). By utilizing some novel methods [14, 22], the time-complexity for building suffix tree and searching similar subsequences can be further improved as O(*n*) and O(*n*+*m*), respectively. Hence, our algorithm is very efficient in terms of time-complexity.

TABLE I
MAIN PARAMETERS

Parameter	Description
SN	The number of symbols
SR(min, max)	The range of length for discovered similar subsequences

```
Input: two transformed symbol sequences S and T,
   SN: number of symbol types
   SR(min, max): length range for similar subsequence
Output: All similar subsequences of S and T.
Method: SimilarSubsequence (S, T, SN, SR (min, max))
1. //Building a suffix tree for sequence S
2. suffixTree := Create a root node with label null;
3. FOR each $suff_i(S)$:=suffix string of S which begins from $i^{th}$
   position
4.    FOR each $suff_j(suff_i(S))$:=suffix string of $suff_i(S)$ which
      start from j position
5.       IF not exists a path from root to leaf in suffixTree
         equaled to $suff_j(suff_i(S))$
6.          Insert the $suff_j(suff_i(S))$ path into suffixTree,
            and record each symbol position of S in each internal
            node of path
7.       ENDIF
8.    ENDFOR
9. ENDFOR
10. //Searching similar subsequence of T in the suffixTree
11. FOR EACH i:= 1 to length(T)
12.    IF ∃ a path starts from root of suffixTree equaled to
       $subseq_{iw}(T)$ /*subsequence of T from $i^{th}$ to $w^{th}$ */
13.       Record the $subseq_{iw}(T)$ as the common subsequence.
14.       PairSet := start position pair ({x},{i}), where {x} is
          the set of the start positions of $subseq_{iw}(T)$ in S
15.       i := w+1;
16.    ENDIF
17. ENDFOR
18. similarSubseqSet := all pairs of similar subsequence,
    PairSet, in S and T by start position
19. similarSubseq := the non-overlapped similar subsequences
    from similarSubseqSet
20. return similarSubseq;
```

Fig. 5. Algorithm for proposed method.

IV. EXPERIMENTAL EVALUATION

To evaluate the performance of the proposed method, we implemented a query system for conducting a series of experiments. The query system takes a time sequence as an input query pattern and returns similar subsequences for all time sequences in the underlying database. All experiments are done on the Windows XP machine with P4-3.0GHz CPU and 1GB main memory. Table I lists the parameters provided by the system for setting up query criteria.

A. Descriptions of Experimental Datasets

In the conducted experiments, we use the Spellman's yeast microarray datasets [21] (http://genome-www.stanford.edu/cellcycle) as tested dataset. The yeast cell cycle datasets contain all genes whose mRNA levels are regulated by the cell cycle. In our experiments, we use the highest-quality time-series dataset, *alpha*, which is with 18 time points.

B. Results on Yeast Cell Cycle Dataset

To evaluate the capability of our method in finding similar subsequences with biological meanings, we take the Filkov's results [7] as a belief base. Filkov *et al.* constructed a database of known regulatory relations in yeast. In the alpha dataset, there are 343 known activations which indicate the relationships of two genes.

We selected the expression profile of gene HTA1 (systematic name as YDR225W) as input query sequence. In Filkov's analysis result, there are several activations related to HTA1. First, we set the parameters as {*SN*=5, *SR*(5,∞)}. Based on the discovered similar subsequences, we found activation relations of four genes with HTA1 as reported in Filkov's study. These activation relations are the following:

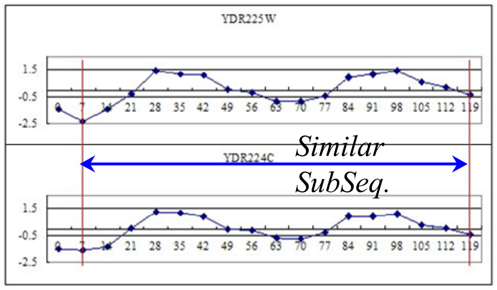
- HTA1 (YDR225W)→HTB1(YDR224C) (Fig. 6(a))
- HTA1 (YDR225W)→HTB2(YBL002W) Fig. 6(b))
- SPT16 (YGL207W)→ HTA1 (YDR225W) (Fig. 6(c))
- SPT21(YMR179W)→ HTA1 (YDR225W) (Fig. 6(d))

The above relations can be validated in the literature [5, 9, 21] and SGD database [19].

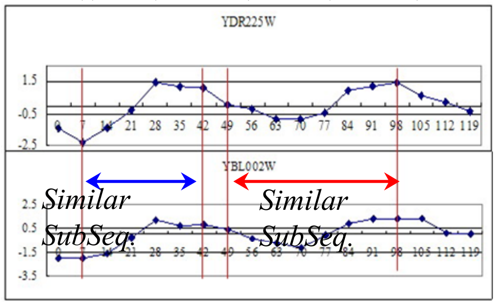
In Fig. 6(d), the PCC similarity of whole sequence between SPT21 and HTA1 is -0.111. However, through our method, two similar subsequences between the two genes can be discovered, that are the subsequences between time point 14 and 70 minutes for SPT21 and that between 35 and 91 minutes for HTA1, respectively. Since the PCC similarity of the two similar subsequences is 0.871, these two genes will be considered to be with potential regulatory relations by our method.

We also compared the difference between our method and those using Euclidean Distance and Pearson Correlation Coefficient (PCC) as similarity measures. The experimental results show that our method can reveal novel gene relations, which can not be discovered by using Euclidean Distance or PCC only. The important relations have been found in traditional biological experiments.

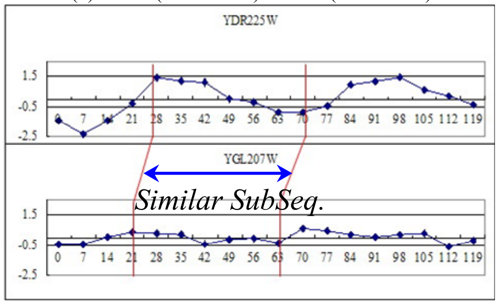
As an example, in considering expression profiles of genes SPT16 (YGL207W) and HTA1 (YDR225W) (in Fig. 7(a)), the similarity value is 4.432 with Euclidean distance and 0.243 with PCC. With these similarity measures, the two genes will be considered as dissimilar. As shown in Fig. 7(a), the dotted lines show the similar subsequences which are in the segments between time point 14 through 70 minutes and 35 through 91



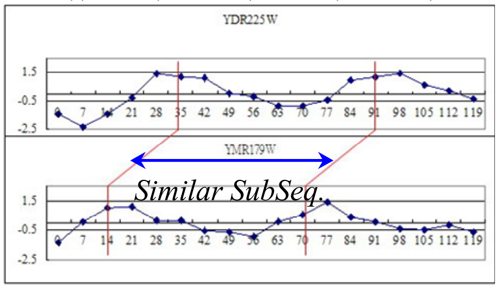
(a)HTA1(YDR225W)→HTB1(YDR224C)



(b) HTA1 (YDR225W)→HTB2(YBL002W)



(c) SPT16 (YGL207W) →HTA1(YDR225W)



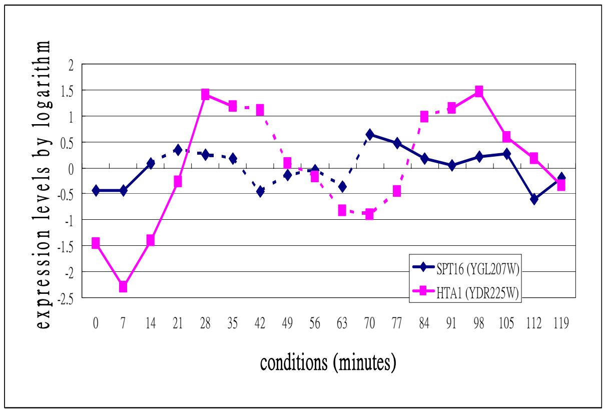
(d) SPT21(YMR179W)→ HTA1 (YDR225W)

Fig. 6. Query result with HTA1(YDR225W) as input.

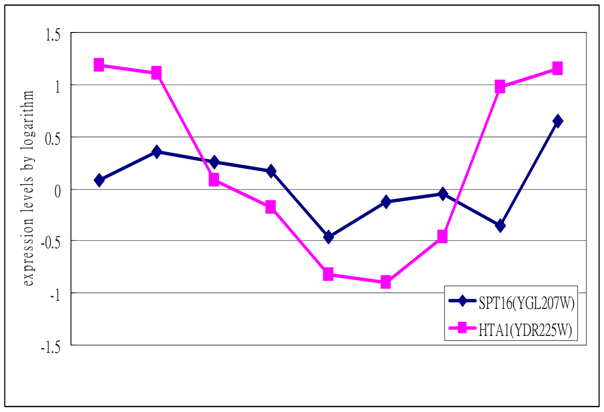
minutes for SPT16 and HTA1, respectively. In fact, the PCC similarity of these two similar subsequences is as high as 0.516.

C. Evaluation of Execution Efficiency

In this experiment, we compare the execution time for the Time-Lagged method [10], Agrawal's method [1] and our method. Fig. 8 shows the performance of the three methods for querying some tested sequence under different settings of subsequence length. By notating Time-Lagged Method, Agrawal's method, and our method as *TL*, *Agr*, and *Ours*, respectively, and using *cTime(M, L)* to denote the execution



(a) Similarity of PCC between SPT16 and HTA1.



(b) The similar subsequences of SPT16 and HTA1.
Fig. 7. Comparisons with different similarity measures.

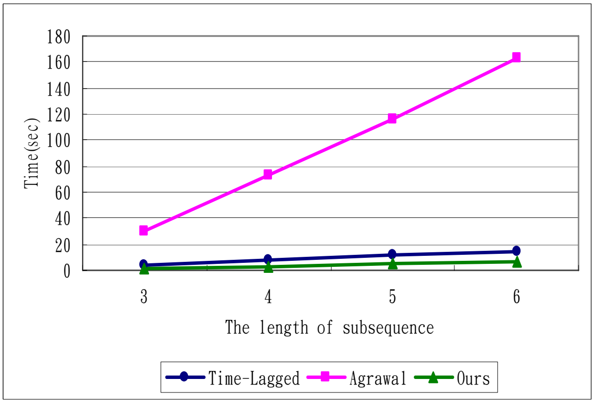


Fig. 8. Comparisons of execution efficiency.

time of method *M* in the subsequence length *L*, the improvement rate of *Ours* over a method *M* is the following.

$$\frac{1}{L\#}\sum_{i=1}^{L\#}\left[1-\frac{cTime(Ours,L_i)}{cTime(M,L_i)}\right] \qquad (5)$$

, where *L#* is the number of subsequence length *L*.

By setting the integral parameter *L* as 3 through 6, the improvement rates are shown to be 57% and 95% for *Ours* over *TL* and *Agr*, respectively. The above results show that our method outperforms Time-Lagged Method and Agrawal's method substantially in terms of execution efficiency.

V. Conclusions

In this paper, we propose an efficient method for mining similar subsequences among time-series and apply it to time-series microarray data for discovering novel gene relations. The proposed method is composed of three main phases, i.e. Angular Transformation, Symbolic Transformation and Similar Subsequence Searching. The main merit of the proposed method is that the four problems of scaling, offset, shift, and noise in time-series analysis are handled simultaneously. Through experimental evaluation on real yeast microarray dataset, our method is shown to be capable of discovering similar subsequences among time-series gene expression profiles efficiently so as to reveal significant functional relations among genes. In the aspect of execution efficiency, our method is also shown to outperform other methods. Hence, our method serves as a promising candidate for efficiently providing useful insights for biologists in discovering novel biological relations among genes. In the future, we would apply our method on more kinds of time-series microarray data for discovering undisclosed relations among genes. Meanwhile, we shall enhance our method to make it more automatic in parameters setting.

Acknowledgment

This research was supported by National Science Council, Taiwan, R.O.C., under grant number NSC 95-2221-E-006-372.

References

[1] Agrawal, R., Lin, K. I., Sawhney, H. S., and Shim, K. "Fast Similarity Search in the Presence of Noise, Scaling, and Translation in Time-Series Databases," *Proceedings of the 21st International Conference on Very Large Data Bases*, Zurich, Switzerland, 1995, pp. 490-501.

[2] Antunes, C. M. and Oliveira, A. L. "Temporal Data Mining: an overview," in *Proc. of the Workshop on Temporal Data Mining, at the 7th International Conference on Knowledge Discovery and Data Mining*, San Francisco, CA, August 2001, pp. 1–15.

[3] Bar-Joseph, Z. "Analyzing time series gene expression data," *Bioinformatics* (20:16), 2004, pp. 2493-2503.

[4] Bickel, D. R. "Probabilities of spurious connections in gene networks: application to expression time series," *Bioinformatics* (21), April 2005, pp. 1121 – 1128.

[5] Compagnone-Post, P. A. and Osley, M. A. "Mutations in the SPT4, SPT5, and SPT6 genes alter transcription of a subset of histone genes in Saccharomyces cerevisiae," *Genetics* (143), 1996, pp. 1543–1554.

[6] Faloutsos, C., Ranganathan, A.M., and Manolopoulos, Y. "Fast Subsequence Matching in Time-Series Databases," *Proceedings of the ACM Special Interest Group On Management of Data*, May 1994, pp. 419-429.

[7] Filkov, V., Skiena, S., and Zhi, J. "Analysis techniques for microarray. time-series data," *Proceedings of the Fifth Annual International Conference on Computational Biology,* Montreal, Canada, 2001, pp. 124-131.

[8] Huang, Y. and Yu, P. S. "Adaptive Query Processing for Time-Series Data," *Proceedings of the 5th International Conference on Knowledge Discovery and Data Mining*, San Diego, CA, Aug. 1999, pp. 15-18.

[9] Iyer, V.R., Eisen, M. B., and Ross, D. T. *et al.* "The Transcriptional Program in the Response of Human Fibroblasts to Serum," *Science* (283), 1999, pp. 83-87.

[10] Ji, L. and Tan, K. L. "Identifying Time-Lagged Gene Clusters on Gene Expression Data," *Bioinformatics* (21:4), Feb. 2005, pp. 509-516.

[11] Kalpakis, K., Gada, D. and Puttagunta, V. "Distance Measures for Effective Clustering of ARIMA Time-Series," *Proceedings of the IEEE International Conference on Data Mining*, San Jose, CA, Nov. 2001, pp. 273-280.

[12] Keogh, E., Lonardi, S. and Chiu, W. "Finding Surprising Patterns in a Time Series Database in Linear Time and Space," *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Edmonton, Alberta, Canada, July 23-26 2002, pp. 550-556.

[13] Lin, J. and Keogh, E. "A Symbolic Representation of Time Series, with Implications for Streaming Algorithms," *ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, San Diego, CA, June 13, 2003.

[14] McCreight, E. M. "A space-economical suffix tree construction algorithm," *Journal of the ACM* (23), 1976, pp. 262-272.

[15] Moon, Y. S., Whang, K. Y. and Han, W. S. "General Match: A Subsequence Matching Method in Time-Series Databases Based on Generalized Windows," *Proceedings of the ACM Special Interest Group On Management of Data Conf.*, 2002, pp. 382-393.

[16] Moon, Y. S., Whang, K. Y., and Loh, W. K. "Duality-Based Subsequence Matching in Time-Series Databases," *Proceedings of the 17th International Conference on Data Engineering*, April 02-06, 2001, pp. 263-272.

[17] Ong, I. M., Glasner, J. D. and Page, D. "Modelling regulatory pathways in E. coli from time series expression profiles" *Bioinformatics* (18**),** Jul 2002, pp. 241 - 248.

[18] Park, S., Kim, S., and Chu, W. W. "Segment-based approach for subsequence searches in sequence databases," *Proceedings of the 16th ACM Symposium on Applied Computing*. Las Vegas, NV, Mar. 11-14, 2001, pp. 248-252.

[19] Saccharomyces Genome Database (SGD): http://www.yeastgenome.org/

[20] Sontag, E., Kiyatkin, A., and Kholodenko, B. N. "Inferring dynamic architecture of cellular networks using time series of gene expression, protein and metabolite data," *Bioinformatics* (**20**), Aug. 2004, pp. 1877–1886.

[21] Spellman, P. T., Sherlock, G., Zhang, M. Q., Iyer, V.R., Anders, K., Eisen, M.B., Brown, P.O., Botstein, D., and Futcher, B. "Comprehensive identification of cell cycle-regulated genes of the yeast saccharomyces cerevisiae by microarray hybridization," *Molecular Biology of the Cell* (9), 1998, pp. 3273-3297.

[22] Ukkonen, E. "On-line construction of suffix trees," *Algorithmica* (14:3), 1995, pp. 249-260.

[23] Vlachos, M., Kollios, G. and Gunopulos, G. "Discovering Similar Multidimensional Trajectories," *Proceedings of the 18th International Conference on Data Engineering*, San Jose, CA, Feb 26-Mar 1, 2002.

[24] Wang, C. and Wang, X. S. "Supporting subseries nearest neighbor search via approximation," *Proceedings of the 9th ACM CIKM Int'l Conference on Information and Knowledge Management,* McLean, VA, Nov 6-11, 2000, pp 314-321.

[25] Wichert, S., Fokianos, K., and Strimmer, K. "Identifying Periodically Expressed Transcripts in Microarray Time Series Data," *Bioinformatics* (20), 2004, pp. 5-20. 2005