Top-k Utility-based Gene Regulation Sequential Pattern Discovery

Morteza Zihayat, Heidar Davoudi, Aijun An
Department of Electrical Engineering and Computer Science
York University, Toronto, Ontario, Canada
Email: \{zihayatm,davoudi,aan\}@cse.yorku.ca

Abstract—Sequential pattern mining has been used in bioinformatics to discover frequent gene regulation sequential patterns based on time course microarray datasets. While mining frequent sequences are important in biological studies for disease treatment, to date, most of the approaches do not consider the importance of the genes with respect to a disease being studied when identifying gene regulation sequential patterns. In addition, they focus on the more general up/down effects of genes in a microarray dataset and do not take into account the various degrees of expression during the mining process. As a result, the current techniques return too many sequences which may not be informative enough for biologists to explore relationships between the disease and underlying causes encoded in gene regulation sequences. In this paper, we propose a utility model by considering both the importance of genes with respect to a disease and their degrees of expression levels under a biological investigation. Then, we design a new method, called TU-SEQ, for identifying top-k high utility gene regulation sequential patterns from a time-course microarray dataset. The evaluation results show that our approach can effectively and efficiently discover key patterns representing meaningful gene regulation sequential patterns in a time-course microarray dataset.

I. INTRODUCTION

Microarrays have been widely used in the biomedical field for discovering differentially expressed genes in human diseases. Many methods have been proposed to monitor massive gene expressions and identify their regulations during a clinical study. However, more and more evidence shows that a human disease cannot be attributed to a single gene but emerges as complex interactions among multiple genetic variants [1]. In recent years, time course gene regulation sequential pattern analysis has become critical in illness events such as cancer formation. Such diseases have to be studied and monitored for a period of time to identify abnormal alternations in gene expressions. However, most of the existing approaches focus on how to discover differentially expressed genes varied with time and they do not consider the associations among these genes. Recently, some studies such as [1], [2], [3], [4] propose to use sequential pattern mining approaches to discover gene regulation sequential patterns.

Sequential pattern mining is an important task in data mining and has been extensively studied by many researchers [5]. Given a dataset of sequences, each containing a list of items/itemsets, sequential pattern mining is to discover sequences of items/itemsets that frequently appear in the dataset. If a potential gene regulation sequential pattern occurs frequently in a period of time in a gene expression dataset, it can be discovered by mining sequential patterns from the dataset.

Although sequential pattern mining has been used to discover gene regulation sequential patterns, some major limitations exist in the previous algorithms. First, these methods mostly choose important sequences based on the frequency/support framework. That is, only gene regulation sequences whose frequency is no less than a user-defined support threshold are chosen as interesting/important gene regulation sequential patterns. However, as clinical studies have shown, the frequency alone may not be informative enough to discover sequences regarding a specific disease. For example, some genes are more important than others in causing a particular disease and some genes are more effective than others in fighting diseases. Moreover, most of the existing approaches consider the more general up/down effects of gene’s behavior (i.e., gene expression) in a microarray dataset by binning the expression value as highly expressed or highly repressed and do not take into account the degree of expressions. For example, a gene may not occur frequently but its behavior is highly remarkable in each appearance or vice versa. As a result, sequences that contain highly important or highly expressed/repressed genes may not be discovered by the frequency-based approaches because they neither consider the importance of genes, nor the various degrees of expression under a biological investigation. To address these limitations of frequency-based mining approaches, a utility is introduced to perform sequential pattern mining. High utility sequential pattern (HUSP) mining aims at extracting valuable and useful sequential patterns from data with respect to an objective. A sequence is a high utility sequential pattern, if its utility, defined based on the objective, in a dataset is no less than a minimum utility threshold. Nonetheless, existing HUSP mining methods are mainly applied to discovering patterns in market basket analysis (e.g., finding profitable customer shopping behavior), and have not been used to find patterns from complex sequential datasets such as time course microarray datasets. In addition, how to define the utility so that it reflects the objective (e.g., a specific disease) effectively and how to convert input sequential dataset (e.g., a time course microarray dataset) to a utility-based sequential database are challenging problems. Second, in gene regulation sequential pattern discovery, it is hard for biologists to determine the
value for the threshold. If the threshold is set too low, a large number of patterns can be found, which is not only time and memory consuming but also makes it hard to analyze the mining results. On the other hand, if the threshold is set too high, there may be very few or even no patterns being found, which means that some interesting patterns are missed. In practice, it is more interesting for biologists to set a bound on the size of output (e.g., top-k patterns), instead of giving a fixed threshold. However, since the threshold is not given in top-k pattern mining, the algorithm essentially needs to start searching for patterns with a very low threshold (e.g., zero or a value close to zero) in order to guarantee that at least k patterns can be found. This incurs very high computational costs. A main challenge is how to increase the threshold without missing any top-k patterns.

To address the above issues, we propose a new algorithm called TU-SEQ (Top-k Utility-based gene regulation sequential pattern discovery) to mine top-k high utility gene regulation sequential patterns (to be defined later) by considering the gene importance and their degrees of expression under a biological investigation. The proposed method only requires specifying a user-desired number k and a disease (as the objective) to explore the k most important gene regulation sequential patterns from a time course microarray dataset. TU-SEQ guarantees that no top-k high utility gene regulation sequences, which take place across different time points during the course of biological observations, will be missed. To the best of our knowledge, our work is the first step towards exploring the impact of both the gene importance and their finer degrees of expression levels under a biological investigation to discover gene regulation sequential patterns. We are hoping this research can shed a light for further research on discovering meaningful patterns from time course microarray datasets. Our contributions are summarized as follows.

- We formulate the problem of top-k utility-based gene regulation sequential pattern discovery. In this regard, we define a utility model by considering both the importance of genes with respect to a disease and their finer degrees of expression under a biological investigation.
- We propose an efficient algorithm, called TU-SEQ, for mining top-k high utility gene regulation sequences from a time course microarray dataset.
- We propose several strategies for initializing and dynamically adjusting the threshold before and during the mining process. The proposed strategies will not miss any top-k patterns.
- We conduct experiments on a real and publicly available time course microarray dataset to evaluate the effectiveness and efficiency of TU-SEQ.
- We develop a web interface\(^1\) to our system. To the best of our knowledge, this is the first demonstration for top-k utility-based gene regulation sequence discovery.

The rest of the paper is organized as follows. Section II summarizes the related work. Section III presents preliminaries and the problem statement. The proposed method is discussed in Section IV. We report our experimental results in Section V. Finally, Section VI concludes the paper.

II. RELATED WORK

Sequential pattern mining has been widely used in the bioinformatics domain for discovering rules for organization of certain elements in genes, for predicting protein function, for analyzing gene expression, for motif discovery in DNA sequences and for discovering sets of genes that are frequently co-expressed in most biological conditions in a microarray dataset. Some of these methods are \textit{apriori algorithm} [6], \textit{half-spaces} [4], and \textit{FPtree algorithm} [7]. Moreover, in [8], a method, called \textit{MAGIIC}, is proposed to discover the structure motifs from protein sequences. In [2], the authors propose an algorithm called \textit{CTGR-Span} (Cross-Timepoint Gene Regulation Sequential pattern) to efficiently discover CTGR-SPs (Cross-Timepoint Gene Regulation Sequential Patterns). However, to the best of our knowledge, all of the aforementioned methods do not consider the objective of the study. That is, the temporal behavior of genes under a biological investigation is ignored in the problem setting, so is the importance of genes with respect to a disease.

Many studies have been also conducted to analyze the association between genes and a specific disease [1]. However, such methods only consider the behavior of each gene individually and they do not take the sequential relationships among genes into account. In [3], the authors propose a method to discover novelty in sequential patterns with respect to a disease (e.g., Alzheimer). However, they do not consider time course sequential databases and also the proposed method still discovers pattern based on frequency.

High utility sequential pattern (HUSP) mining has been studied recently [9], [10], [11], [12], [13], [14]. The concept of HUSP mining was first proposed by Ahmed et al [9]. They proposed two algorithms, called UL and US, for mining HUSPs. UL is a level-wise candidate generation-and-testing algorithm and US is a pattern growth method inspired byPrefixSpan [15]. Yin et al [12] proposed the USpan algorithm for mining HUSPs. In this study, a lexicographic tree was used to extract the complete set of high utility sequential patterns and designed mechanisms for expanding the tree with two pruning strategies. In [14], we proposed a single-pass algorithm to find high utility sequential patterns in a dynamic data stream environment. We also showed that \textit{HUSP-Stream} outperforms USpan in terms of run time and memory usage. However, most of the mentioned methods do not find patterns from complex sequential datasets such as time course microarray datasets.

So far, no study has been conducted to learn \textit{utility-based gene regulation sequential patterns} in a time course microarray dataset, which is more challenging than finding frequent time course gene regulation sequential patterns.

\(^1\)Demo available at http://graph.cse.yorku.ca:8080/GeneAssociation/
TABLE I
(A) AN EXAMPLE OF A TIME COURSE MICROARRAY DATASET, (B) FOLD CHANGES OF GENE/PROBE VALUES

\[
\begin{array}{cccccc}
\text{Patient ID} & G_1 & G_2 & G_3 & G_4 & G_5 \\
\hline
r_1 & 200 & 146 & 100 & 50 & \\
& 300 & 254 & 159 & 127 & \\
& 400 & 356 & 251 & 243 & \\
r_2 & 500 & 356 & 159 & 243 & \\
& 600 & 456 & 356 & 243 & \\
& 700 & 546 & 456 & 345 & \\
r_3 & 800 & 646 & 546 & 456 & \\
& 900 & 746 & 646 & 546 & \\
& 1000 & 846 & 746 & 646 & \\
\end{array}
\]

III. Definitions and Problem Statement

A. A Time course Microarray dataset and its transformation to a time course sequential dataset

A time course microarray dataset cannot be used directly to mine high utility gene regulation sequential patterns. In this section, we describe how to convert a time course microarray dataset to a proper time course sequential dataset.

Table I(a) shows an example of time course microarray dataset obtained from a biological investigation which consists of three patients whose IDs are $P_1$, $P_2$ and $P_3$. In this table, the gene expression values of three genes $G_1$, $G_2$ and $G_3$ are presented over four time point samples $TS_1$, $TS_2$, $TS_3$ and $TS_4$.

In each time sample, each gene has a **temporal behavior** which is expressed by a real value. We consider the first time sample as a baseline to derive the temporal behavior of each gene at each time sample. That is, the **temporal behavior** of a gene at a time sample $TS$ is the expression value of the gene at $TS$ divided by the expression value of the gene at the first time sample. It represents the degree of expression of the gene at time sample $TS$. Table I (b) shows the temporal behavior values as a fold change matrix.

Given the fold change matrix and a threshold $\gamma$, each expression value in the dataset is transformed as *up-regulated* (representing by $+$ meaning that the value is greater than $\gamma$), *down-regulated* (representing by $-$ meaning that the value is less than $-\gamma$), or *normal* (neither expressed nor repressed) and only the gene expressions that are up-regulated or down-regulated are preserved. Each gene ($G_x$) in a sample can be thought of as being one of two *items*, one item referring to the gene being up (i.e., $G_{x^+}$), the other referring to the gene being down (i.e., $G_{x^-}$).

Given $\gamma = 1.5$, Table II(a) shows the converted dataset (i.e., the time course sequential dataset). For example, in patient $P_3$, up-regulated $G_{1^+}(2.2)$ and down-regulated $G_{2^-}(3.2)$ are considered to occur at the same time (i.e., $TS_2$), where 2.2 and 3.2 are the temporal behavior of $G_{1^+}$ and $G_{2^-}$ (as defined above) respectively.

\[\text{TABLE II}
\]

\[\text{(a) A time course sequential dataset from time course microarray dataset in Table I(a), (b) Gene importance table}
\]

For example, in Table II(b), given gene $G_{1^-}$ and time sample $TS_3$ in sequence $P_1$, $IGU(G_{1^-}, P_1^3) = 2.4$. This value specifies the relative abundance of the gene in the time sample.

B. Definitions

Let $G = \{G_{1^+}, G_{1^-}, G_{2^+}, G_{2^-}, ..., G_{n^+}, G_{n^-}\}$ be a set of distinct gene regulation items. A *geneset* $GS$ is a set of gene regulation items. A *time-course sequential dataset* is a set of patients $(P_1, P_2, ..., P_K)$, where each patient has a patient identifier $P_r$ and consists of an ordered list of time point samples (or in brief time samples (TSs)) where each $TS$ is a geneset. The time sample $TS_d$ for patient $P_r$ is denoted as $P_r^d$.

**Definition 1**: The **importance of gene** $g$ is a score which is calculated based on one or more disease-related variables $var_1, var_2, ..., var_k$ which is defined as follows. $GI_1(g) = f_g(var_1, var_2, ..., var_k)$, where $f_g$ is the function for calculating the importance of $g$.

For example, Table II(b) shows the importance of genes with respect to a disease. In this work, the importance of $G_x$ represents the importance of both $G_{x^+}$ and $G_{x^-}$ gene items.

**Definition 2**: **Internal utility** or temporal behavior of a gene $g$ is a real value assigned to $g$ in the time sample $TS_d$ of patient $P_r$ (i.e., $P_r^d$). It is denoted as $IGU_{dis}(g, P_r^d)$ and is defined as the expression value of $g$ at $TS_d$ divided by the expression value of $g$ at the first time sample in $P_r$. For example, in Table II(a), given gene $G_{1^-}$ and time sample $TS_3$ in sequence $P_1$, $IGU(G_{1^-}, P_1^3) = 2.4$. This value specifies the relative abundance of the gene in the time sample.

**Definition 3**: (Utility of gene $g$ in time sample $P_r^d$) Given gene $g$ and time sample $P_r^d$, **gene utility** is defined as a combination of gene importance and internal utility of $g$ w.r.t. disease $dis$ as follows. $GU(g, P_r^d) = f_{gu}(GI_1(g), IGU(g, P_r^d))$, where $f_{gu}$ is the function for calculating utility.

For simplicity, we define the $f_{gu}$ function as $f_{gu}(GI_1(g), IGU(g, P_r^d)) = GI_1(g) \cdot IGU(g, P_r^d)$.

**Definition 4**: The **utility of a geneset GS in a time sample TS_d** of a patient $P_r$ where $GS \subseteq TS_d$, is defined as $GU(GS, P_r^d) = \sum_{g \in GS} GU(g, P_r^d)$.

**Definition 5**: (Occurrence of a sequence $\alpha$ in a patient $P_r$) Given a patient $P_r = (P_r^1, P_r^2, ..., P_r^n)$ and a gene regulation sequence $\alpha = (GS_1, GS_2, ..., GS_Z)$ where $P_r^g$ is a time sample and $GS_g$ is a geneset, $\alpha$ occurs in $P_r$ iff there exist integers $1 \leq e_1 < e_2 < ... < e_Z \leq n$ such that
GS1 ⊆ P^e_1, GS2 ⊆ P^e_2, ..., GS_Z ⊆ P^e_Z. The ordered list of genesets \( \{P^e_1, P^e_2, ..., P^e_Z\} \) is called an occurrence of \( \alpha \) in \( P_r \). The set of all occurrences of \( \alpha \) in \( P_r \) is denoted as \( \text{OccSet}(\alpha, P_r) \).

**Definition 6:** The (utility of a gene regulation sequential pattern \( \alpha \) in a patient sequence \( P_r \)) Let \( \tilde{\delta} = \langle P^e_1, P^e_2, ..., P^e_x \rangle \) be an occurrence of \( \alpha = \langle GS_1, GS_2, ..., GS_Z \rangle \) in the sequence \( P_r \). The utility of \( \alpha \) w.r.t. \( P_r \) is defined as \( GU(\alpha, \tilde{\delta}) = \sum_{i=1}^{Z} GU(GS_i, P^e_i) \). The utility of \( \alpha \) in \( P_r \) is defined as \( GU(\alpha, P_r) = \max\{GU(\alpha, \tilde{\delta}) \mid \tilde{\delta} \in \text{OccSet}(\alpha, P_r)\} \).

**Definition 7:** The (utility of a gene regulation sequence \( \alpha \) in a time course sequential dataset \( D \)) The utility of a gene regulation sequence \( \alpha \) in a time course sequential dataset \( D \) is defined as \( GU(\alpha, D) = \sum_{P_r \in D} GU(\alpha, P_r) \).

**Definition 8:** (High Utility Gene Regulation Sequence (HUGS)) Given a threshold \( \delta \), a sequence \( \alpha \) is a High Utility Gene Regulation Sequence (HUGS) in a time course sequential dataset \( D \), if \( GU(\alpha, D) \) is no less than \( \delta \).

**Definition 9:** (Top-k High Utility Gene Regulation Sequence in a time course sequential dataset \( D \)) A gene regulation sequence \( \alpha \) is called a top-k High Utility Gene Regulation Sequence (HUGS) in \( D \), if there are less than \( k \) sequences whose utility value in \( D \) is no less than \( GU(\alpha, D) \).

**Problem Statement.** Given a time course sequential dataset \( D \) and a user-defined number \( k \), the problem of finding the complete set of top-k high utility gene regulation sequential patterns in \( D \) is to discover all the gene regulation sequential patterns whose utility is no less than \( \text{minUtil}_{\text{opt}} \), where \( \text{minUtil}_{\text{opt}} = \min\{GU(\beta, D) \mid \beta \in \text{THUGS}_D \} \), where \( \text{THUGS}_D \) is the set of top-k HUGSs over \( D \).

**IV. Top-k Utility-based Gene Regulation Sequential Pattern Discovery**

In this section, we propose an efficient algorithm called **TU-SEQ** (Top-k Utility-based gene regulation SEQuential pattern discovery) to find top-k HUGSs without specifying the minimum threshold. First, a basic approach called **TU-SEQ\_Base** is presented. Later, we present a novel strategy for initializing the threshold with respect to the given \( k \) in **TU-SEQ\_Base**.

A. **TU-SEQ\_Base** approach

The proposed baseline approach **TU-SEQ\_Base** takes \( k \) as an input parameter and returns top-k sequences with the highest utilities in a time course sequential dataset \( D \). It is an extension of **HUSP-Stream**, our recent proposed method for mining high utility sequential patterns [14], and it applies the idea of **ItemUtilLists** and **HUSP-Tree** to maintain the information of potential top-k HUGSs. **HUSP-Stream** is a threshold-based approach and is not able to discover top-k HUGSs.

We first briefly describe **ItemUtilLists** and **HUSP-Tree**. For more details about the data structures, readers can refer to [14].

**ItemUtilLists** is a vertical representation of the time samples in the dataset. The **ItemUtilLists** of a gene \( G \) consists of several tuples. Each tuple stores the utility of gene \( G \) in the time sample \( P^u \) (i.e., time sample \( TS_p \) in patient \( P_r \)) that contains \( G \). Each tuple has three fields: \( P^ID \), \( TID \) and \( \text{util} \). Fields \( P^ID \) and \( TID \) store the identifiers of \( P_r \) and \( TS_p \), respectively. Field \( \text{util} \) stores the utility of \( G \) in \( P^u \) (Definition 3). Figure 1(a) shows the **ItemUtilLists** of \( G_{1+} \), \( G_{2-} \) and \( G_{3–} \) in Table II(a).

A **HUSP-Tree** is a lexicographic sequence tree where each non-root node represents a sequence of genesets. Figure 1(b) shows part of the **HUSP-Tree** for the the dataset in Table II(a), where the root is empty. Each node at the first level under the root represents a sequence of length 1, a node on the second level represents a 2-sequence, and so on. Each non-root node of a **HUSP-Tree** is designed to have a field, called **SeqUtilList**, for maintaining information about the sequence represented by a node. The sequence utility list (**SeqUtilList**) of a sequence \( \alpha \) is a list of three-value tuples. Each tuple \( P^ID \), \( TID \), \( \text{util} \) represents an occurrence of \( \alpha \) in a sequence of the dataset and the utility of \( \alpha \) with respect to the occurrence. The \( P^ID \) in a tuple is the ID of the patient in which \( \alpha \) occurs, \( TID \) is the ID of the last time sample in the occurrence of \( \alpha \), and \( \text{util} \) is the utility value of \( \alpha \) with respect to the occurrence. We denote the **SeqUtilList** of \( \alpha \) as **SeqUtilList(\alpha)**.

A non-root node in a **HUSP-Tree** is either **I-node** or **S-node**.

**Definition 10:** (I-concatenate Sequence) Given a sequence pattern \( \alpha \), an I-concatenate pattern \( \beta \) represents a sequence generated by adding a gene \( G \) into the last geneset of \( \alpha \) (denoted as \( \alpha \oplus G \)). A node whose pattern is an I-concatenate sequence is called **I-node**.

**Definition 11:** (S-concatenate Sequence) Given a sequence \( \alpha \), an S-concatenate pattern \( \beta \) represents a sequence generated by adding a geneset \( \{G\} \) after the last geneset of \( \alpha \) (denoted as \( \alpha \ominus G \)). A node whose pattern is a S-concatenate sequence is called **S-node**.

In Figure 1(b), the node for sequence \( \{G_{1+}G_{3+}\} \) is an **I-node**, while the node for \( \{G_{1+}\} \) is a **S-node**.

**HUSP-Tree Construction:** A **HUSP-Tree** is constructed recursively in a top-down fashion using **ItemUtilLists**. The first level of the tree under the root is constructed by using the genes in **ItemUtilLists** as nodes. The **SeqUtilList** of these nodes is the **ItemUtilList** of the items. Given a non-root node, its child nodes are generated using **I-Step** and **S-Step**, which generate **I-nodes** and **S-nodes** respectively. We demonstrate **I-Step** and **S-Step** procedures of pattern \( \alpha = \{G_{1+}\} \) with sequence \( P_2 \) in Table II (a). We start from the **I-Step**. Given the pattern \( \alpha \) and gene \( G = G_{2–} \), in order to form \( \beta = \{G_{1+}G_{2–}\} \) and calculate its utility, **I-Step** is applied as follows. According to Table II (a), only time sample \( P^u_2 \) has \( G_{2–} \) can be used to form sequence \( \beta \). The utility of \( \beta \) is \( GU(\alpha, P^u_2) \) plus the newly added gene’s utility which is \( GU(G_{2–}, P^u_2) \). Therefore, \( GU(\beta, P^u_2) = 6.1 \times 0.8 + 3.2 \times 0.6 = 4.88 + 1.92 = 6.8 \). Given pattern \( \alpha = \{G_{1+}G_{2+}\} \) and \( G = G_{3–} \), to construct pattern \( \beta = \{G_{1+}G_{2+}\} \{G_{3–}\} \) and calculate its utility, **S-Step** works as follows. Since geneset \( \{G_{3–}\} \) must occur in any
time samples after \( \alpha \) occurs, the only case for \( \{G_3^+\} \) is in \( P_{1}^4 \). Hence, \( GU(\beta, P_2) = \{6.8 + 0.2\} = 7 \).

In addition to ItemUtilLists and HUSP-Tree, T-HUSP\(_{Base}\) engages a structure called TKList to maintain the information of top-k high utility gene regulation sequential patterns.

**Definition 12:** Top-k HUGS List (TKList) is a fixed-size sorted list which maintains the top-k high utility gene regulation sequential patterns and their utility values. Each tuple in TKList has two elements: \((\alpha, \text{util})\), where \( \alpha \) is the pattern and \( \text{util} \) is the utility of pattern \( \alpha \) in the dataset.

Since the threshold is not given as an input parameter, TU-SEQ\(_{Base}\) employs a variable called \( \text{minUtil} \) which is the current threshold and is set to zero at the beginning. This variable is used to prune unpromising candidates during the mining process.

Given a time course sequential dataset \( D \) and \( k \), TU-SEQ\(_{Base}\) finds top-k HUGSs as follows. TU-SEQ\(_{Base}\) first sets \( \text{minUtil} \) to 0. Then, it constructs ItemUtilList and HUSP-Tree by applying the S-Step and I-Step procedures. As soon as a new node is added to HUSP-Tree, the pattern represented by the node and its utility are added as a new tuple to TKList. Once \( k \) valid patterns are found, the \( \text{minUtil} \) is raised to the utility value of the pattern with the lowest utility value in TKList. Raising the \( \text{minUtil} \) value is used to prune the search space when searching for more patterns. Thereafter, whenever a new node is inserted to the tree, its pattern is added to TKList. Then, the patterns in TKList whose utility is less than \( \text{minUtil} \) are removed from TKList, and \( \text{minUtil} \) is updated by the utility value of the \( k \)th pattern in TKList. TU-SEQ\(_{Base}\) continues constructing HUSP-Tree and finding more patterns until no node can be generated, which means that it has found the top-k HUGSs in the dataset.

Since HUSP-Stream is correct and complete [14], it is clear that TU-SEQ\(_{Base}\) is correct and will not miss any top-k HUGSs.

**B. PES (Pre-Evaluation using genes and sequences) Strategy**

Although TU-SEQ\(_{Base}\) correctly discovers the top-k high utility gene regulation sequential patterns in the dataset, it generates too many invalid sequence candidates since \( \text{minUtil} \) starts from 0. This directly degrades the performance of the mining task. To address this problem, we propose an effective strategy for initializing the threshold before HUSP-Tree construction to improve the performance. The development of the proposed strategy is based on the following lemmas.

**Lemma 1:** Given a time course sequential dataset \( D \), let \( L = \{\alpha_1, \alpha_2, ..., \alpha_m\} \) be a set of gene regulation sequences \((m \geq k)\), where \( \alpha_i \) is the sequence with the \( i \)-th highest utility value in \( L \). For any sequence \( \beta \), if \( GU(\beta, D) < GU(\alpha_k, D) \), \( \beta \) is not a top-k high utility gene regulation sequence.

**Rationale.** According to Definition 9, if there exist \( k \) sequences with utility values higher than the utility of \( \beta \), \( \beta \) is not a top-k high utility gene regulation sequential pattern.

**Lemma 2:** Let \( L = \{\alpha_1, \alpha_2, ..., \alpha_m\} \) be a set of sequences \((m \geq k)\), where \( \alpha_i \) is the \( i \)-th sequence in \( L \). If \( \forall i < j, GU(\alpha_i, D) \geq GU(\alpha_j, D) \) and \( H_3 \) be the set of high utility gene regulation sequential patterns when the threshold is \( \delta \). If \( \text{minUtil} = GU(\alpha_k) \), then \( H_{\text{minUtil}} \subseteq H_{\text{minUtilopt}} \).

**Rationale.** Let \( H \) be the complete set of top-k high utility gene regulation sequential patterns. If \( |H| \geq k \), \( \text{minUtilopt} = \min\{GU(\alpha, D)\} \alpha \in H \). Since \( \text{minUtilopt} = \min\{GU(\alpha, D)\} \alpha \in H \). Hence, \( \text{minUtilopt} \geq \min\{GU(\alpha_i)\} \alpha_i \in L, 1 \leq i \leq k \} = GU(\alpha_k, D) = \text{minUtil} \), then \( \text{minUtilopt} \geq \text{minUtil} \) and \( H_{\text{minUtilopt}} \subseteq H_{\text{minUtil}} \).

According to the above lemmas, PES initializes TKList by inserting the utility of genes and sequences in the dataset to the TKList before the tree construction. After all time samples in the dataset \( D \) are successfully inserted to ItemUtilList, PES calculates the utility of each gene and each sequence. Given the updated TKList, \( \text{minUtil} \) is initialized by the utility value of \( k \)th pattern in TKList.

**Example 1:** Given \( k = 4 \), the time course sequential dataset \( D \) in Table II(a), the utility of gene \( G_{1}^+ \) in \( D \) is calculated as follows: \( GU(G_{1}^+, D) = GU(G_{1}^+, P_1) + GU(G_{1}^+, P_2) + GU(G_{1}^+, P_3) = 1.76 + 4.88 + 0 = 6.64 \). The utility values of other genes in \( D \) can be calculated similarly. \( GU(G_{1}^-, D) = 15.84 \), \( GU(G_{2}^+, D) = 3.72 \), \( GU(G_{2}^-, D) = \).
Algorithm 1 TU-SEQ

| Input: a time-course sequential dataset D, k |
| Output: THUGSD |

1: ItemUtilLists, TKList ← ∅
2: for each time sample $P^t_1 \in D$ do
3:   for each gene $G \in P^t_1$ do
4:      insert $⟨r, t, GU(G, P^t_1)⟩$ to ItemUtilLists($G$)
5:   end for
6: end for
7: Initialize TKList and minUtil using PES strategy.
8: Construct HUSP-Tree using ItemUtilLists and minUtil
9: Update minUtil whenever a new node is added to the tree
10: if the user requests to get top-k HUGSs then
11:   THUGSD ← all the patterns and their $util$ values stored in TKList
12: end if
13: Return THUGSD if requested

7.32, $GU(G_3, D) = 0.51$ and $GU(G_2, D) = 0.35$. The utility of each sequence can be easily calculated using $ItemUtilLists$. After $D$ is processed, $P_1$ and its utility (e.g., 11.5) is inserted into the TKList. Similarly, the other sequences in $D$ (e.g., $P_2$ (12.18) and $P_3$ (25.07)) are scanned and then they are inserted into the TKList. With the three sequences, six genes and their utility values in $D$, the utility values in the TKList are [25.7, 15.84, 12.18, 11.5]. Hence, $minUtil = 11.5$ after applying PES strategy.

As seen from the example, the PES strategy effectively raises the minimum threshold to a reasonable level before the tree construction, and prevents from generating unpromising candidates.

C. Overview of TU-SEQ

The overview of TU-SEQ is presented in Algorithm 1. Given a time course sequential dataset $D$, TU-SEQ first constructs the ItemUtilLists for storing the information for every gene in each time sample in $D$. Then, it initializes TKList and minUtil by applying PES strategy based on genes in the ItemUtilLists and the sequences in the dataset. Then, TU-SEQ constructs HUSP-Tree using I-Step and S-Step. During the tree construction, whenever a new node is added to the tree, TKList and minUtil are updated as explained in section IV-A. Finally, if the user requests to find top-k HUGSs from the dataset, TU-SEQ returns all the patterns and their $util$ values in the TKList as top-k HUGSs (i.e., THUGSD).

V. EXPERIMENTAL RESULTS

In this section, the proposed method for finding top-k high utility gene regulation sequential patterns is evaluated. All the algorithms are implemented in Java. The experiments are conducted on an Intel(R) Core(TM) i7 2.80 GHz computer with 12 GB of RAM. The GSE6377 dataset [16], downloaded from the GEMMA database, is used in our experiments. McDunn et al. [16] attempted to detect 8,793 transcriptional changes in 11 ventilator-associated pneumonia patients leukocytes across 10 time samples.

A. The importance of genes with respect to pneumonia

Several databases have been developed providing associations between genes and diseases such as CTD [17]. Each of these resources considers different aspects of the phenotype-genotype relationship and they are not complete. Based on our investigation, DisGeNET$^4$ is a discovery platform which integrates different databases with information extracted from the literature to create a comprehensive view of the state of the art knowledge within this research field. In this paper, we consider the score proposed by DisGeNET$^5$ as the importance of each gene with respect to the disease. This score considers several variables such as number and type of sources (level of curation, model organisms) and the number of publications supporting the association to rank genes with respect to a specific disease. Table III shows top-20 genes w.r.t. Pneumonia and their scores.

We calculate the utility$^6$ of gene $G$ in time sample $P^d_t$ as follows: $GU(G, P^d_t) = GI(G) \times IGU(G, P^d_t)$, where $GI(G)$ is the importance of $G$ w.r.t. Pneumonia retrieved from DisGeNET and $IGU(G, P^d_t)$ is the expression value of $G$ in time sample $TS_d$ in sequence $P_t$.

B. HUGSs comparison with FGSs

In this section, we address whether patterns discovered by TU-SEQ contain potential genes/regulations which have not been reported in previous literature yet. We first run TU-SEQ to extract top-k HUGSs with respect to Pneumonia. We also run a recent method called CTGR-Span [2], to discover frequent gene regulation sequential patterns (i.e., FGSs) from the dataset. Given a gene regulation sequential pattern $\alpha$ and a disease $dis$, we evaluate the quality of the results using popularity of a sequence [18] which is defined as follows $Pop(\alpha, dis) = \frac{\sum_{i \in [\alpha]} w(i, dis)}{|\alpha|}$, where $w(i, dis)$ is the importance of the popular gene $i$ for disease $dis$. Without loss of generality, we consider the genes presented in Table III as popular genes and $w(i, dis) = 20 - rank(i, dis) + 1$. For the genes which are not presented in the list, $w(i, dis) = 1$.

Table IV shows top-4 HUGSs extracted by TU-SEQ and top-4 FGSs extracted by CTGR-Span, sorted by the utility and support respectively. Table IV suggests that the frequent sequences are not necessarily popular w.r.t. the disease even though their support value is high. This is due to the fact that these patterns are discovered based on their frequency in the dataset which is not informative enough. On the other hand, TU-SEQ returns the patterns whose popularity is relatively high. These patterns help biologists select relevant sequences regarding a specific disease and also identify the relationships between important genes and the other genes.

Table V shows the average value of utility (i.e., $GU$). $Pop$ and $Sup$ for top-1000 patterns returned by the methods. Given sequence $\alpha$, the last two columns present harmonic mean of

---

$^4$http://www.disgenet.org/web/DisGeNET/menu

$^5$http://www.disgenet.org/web/DisGeNET/menu

$^6$The model can be plugged in as desired. The use of more sophisticated the model may further improve the quality of the results.
In this section, we evaluate the performance of the algorithms using the following measures: (1) Run Time (sec.): the total execution time of the algorithms. and (2) Memory Usage (MB): the average memory consumption per window.

Since there is no known algorithm can solve the problem of mining top-k utility-based gene regulation sequential patterns, we thus compare TU-SEQ with our proposed baseline approach (i.e., TU-SEQ\textsubscript{Base}) as described in subsection IV-A. We also use the threshold-based approach (i.e., HUSP-Stream) proposed in [14] as another baseline approach. After getting the utility of the k-th pattern, that is the optimal minimum threshold in Definition 9, we use this value as the threshold for running the threshold-based method.

We compare TU-SEQ with TU-SEQ\textsubscript{Base} and HUSP-Stream on the GSE6377 dataset. The run time of mining top-k high utility gene regulation sequential patterns by the methods is presented in Figure 2(a). The results show that TU-SEQ is more than 5 times faster than TU-SEQ\textsubscript{Base}. For larger values of k, TU-SEQ\textsubscript{Base} cannot finish the mining in 12+ hours. Besides, the gap between TU-SEQ and TU-SEQ\textsubscript{Base} increases with the increase of k. The results indicate that the proposed strategy (i.e., PES strategy) is effective for top-k pattern mining.

The memory consumption of the algorithms on the dataset is shown in Figure 2(b). It can be seen that TU-SEQ uses less memory than TU-SEQ\textsubscript{Base}. The reason is that TU-SEQ creates a smaller search space because it applies the proposed strategy, thus increases the threshold quicker than TU-SEQ\textsubscript{Base}. Since all the methods use similar approach to construct the tree, the main factor in memory consumption is the threshold used by each of them. HUSP-Stream uses the optimal threshold (i.e., \textit{minUtil\textsubscript{opt}}), hence it prunes the search space efficiently and its memory usage is less than the other methods.

## TABLE III

### TOP-20 GENES RELATED TO PNEUMONIA

<table>
<thead>
<tr>
<th>Rank</th>
<th>Gene (GU)</th>
<th>Rank</th>
<th>Gene (GU)</th>
<th>Rank</th>
<th>Gene (GU)</th>
<th>Rank</th>
<th>Gene (GU)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>CAT</td>
<td>6</td>
<td>SPLPC</td>
<td>10</td>
<td>FGF2</td>
<td>15</td>
<td>HNRG1</td>
</tr>
<tr>
<td>2</td>
<td>FGF2</td>
<td>7</td>
<td>FGF3</td>
<td>11</td>
<td>FGF2A</td>
<td>16</td>
<td>HNRG1</td>
</tr>
<tr>
<td>3</td>
<td>FGF4</td>
<td>8</td>
<td>FGF5</td>
<td>12</td>
<td>FGF2B</td>
<td>17</td>
<td>HNRG1</td>
</tr>
<tr>
<td>4</td>
<td>FGF6</td>
<td>9</td>
<td>FGF7</td>
<td>13</td>
<td>FGF2C</td>
<td>18</td>
<td>HNRG1</td>
</tr>
<tr>
<td>5</td>
<td>FGF8</td>
<td>10</td>
<td>FGF9</td>
<td>14</td>
<td>FGF2D</td>
<td>19</td>
<td>HNRG1</td>
</tr>
</tbody>
</table>

## TABLE IV

### TOP-4 HUGS\textsubscript{S} VERSUS TOP-4 FGSS WITH RESPECT TO SUPPORT AND GDA

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>ID</th>
<th>Sequence of genes (e.g., a)</th>
<th>Support</th>
<th>Utility</th>
</tr>
</thead>
<tbody>
<tr>
<td>TU-SEQ</td>
<td></td>
<td>HUGS\textsubscript{1} (CAT) (CAT MBL2) (CAT)</td>
<td>9</td>
<td>250600</td>
</tr>
<tr>
<td></td>
<td></td>
<td>HUGS\textsubscript{2} (GOLPH3 PDPN) (CAT) (PDPN) (CAT)</td>
<td>9</td>
<td>250325</td>
</tr>
<tr>
<td></td>
<td></td>
<td>HUGS\textsubscript{3} (CAT MBL2) (CAT) (CAT)</td>
<td>9</td>
<td>249741</td>
</tr>
<tr>
<td></td>
<td></td>
<td>HUGS\textsubscript{4} (PDPN) (CAT) (PDPN) (CAT) (PDPN)</td>
<td>9</td>
<td>243037</td>
</tr>
<tr>
<td>CTGR-Span</td>
<td></td>
<td>FGS\textsubscript{1} (LCN2 S100A12) (LCN2)</td>
<td>11</td>
<td>59981</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FGS\textsubscript{2} (LCN2) (S100A12) (CSF3) (LCN2)</td>
<td>11</td>
<td>59962</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FGS\textsubscript{3} (CSF3 S100A12)</td>
<td>11</td>
<td>59931</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FGS\textsubscript{4} (LCN2 S100A12) (LCN2 S100A12)</td>
<td>11</td>
<td>58514</td>
</tr>
</tbody>
</table>

## TABLE V

### THE AVERAGE VALUE OF Sup, GU, Pop, GU-Pop AND Sup-Pop FOR TOP-1000 SEQUENCES RETURNED BY THE METHODS

<table>
<thead>
<tr>
<th>Method</th>
<th>Sup</th>
<th>GU</th>
<th>Pop</th>
<th>GU-Pop</th>
<th>Sup-Pop</th>
</tr>
</thead>
<tbody>
<tr>
<td>TU-SEQ</td>
<td>5</td>
<td>198939</td>
<td>12.5</td>
<td>24.96</td>
<td>7.32</td>
</tr>
<tr>
<td>CTGR-Span</td>
<td>10</td>
<td>44691</td>
<td>1.02</td>
<td>2.05</td>
<td>1.86</td>
</tr>
</tbody>
</table>

(GU, Pop) and (Sup, Pop) which are calculated as follows: 
\[
GU = Pop = 2 \times GU \times Pop, \quad \text{Sup} = Pop = 2 \times Sup \times Pop
\]

According to Table V, the higher values of these measures for TU-SEQ show that even though the patterns returned by TU-SEQ are not as frequent as those of returned by CTGR-Span, they are not only much more relevant to the disease, but also they are frequent enough.

## C. Efficiency of TU-SEQ

In this section, we evaluate the performance of the algorithms using the following measures: (1) Run Time (sec.): the total execution time of the algorithms. and (2) Memory Usage (MB): the average memory consumption per window.

Since there is no known algorithm can solve the problem of mining top-k utility-based gene regulation sequential patterns, we thus compare TU-SEQ with our proposed baseline approach (i.e., TU-SEQ\textsubscript{Base}) as described in subsection IV-A. We also use the threshold-based approach (i.e., HUSP-Stream) proposed in [14] as another baseline approach. After getting the utility of the k-th pattern, that is the optimal minimum threshold in Definition 9, we use this value as the threshold for running the threshold-based method.

We compare TU-SEQ with TU-SEQ\textsubscript{Base} and HUSP-Stream on the GSE6377 dataset. The run time of mining top-k high
D. Demonstration

We also develop a web interface\(^7\) to our system using Java. To the best of our knowledge, this is the first demonstration for top-k high utility gene regulation sequential pattern discovery. In order to evaluate TU-SEQ to find top-k HUGSs from a time course gene sequential dataset, we mine the GSE6377 dataset. According to DisGNET, Athma and Rheumatoid Arthritis are among top 10 diseases that share genes with Pneumonia. Hence, in our demonstration, we also present top-k high utility gene regulation sequential patterns with respect to Athma and Rheumatoid Arthritis. Moreover, the patterns discovered by CTGR-Span [2] are presented. In the first page of the interface, the user can specify disease, ranking measure, number of output gene regulation sequences (i.e., \(k\)) and discovery method(s).

In the demonstration, users are able to compare the algorithms in the following aspects:

1) **Meaningful results:** Our method based on the utility model produces more meaningful sequences than the other method.

2) **Top-k gene regulation sequential patterns presentation:** The sequences retrieved by the methods are provided in a meaningful graphical presentation.

3) **Additional information:** Additional information such as values for the other measures than the selected one for ranking and top-20 genes related to the selected disease.

VI. CONCLUSION

In this paper, we defined the problem of top-k utility-based gene regulation sequential pattern discovery to find patterns with stronger meanings in biology. By solving this problem, we addressed the limitations of previous frequency-based gene regulation sequential pattern mining methods. We first proposed a utility model by considering the importance of genes with respect to a disease and their temporal behavior. Then, using the utility model, we proposed an efficient algorithm called TU-SEQ to find top-k high utility gene regulation sequential patterns. Our experiments suggested that TU-SEQ is much more efficient and scalable than baseline algorithms for top-k high utility gene sequential pattern discovery. We also showed that TU-SEQ is an effective tools to provide biologists with further insights into the relationships of gene regulatory events and interactions in biological studies with respect to a specific disease.

\(^7\)http://graph.cse.yorku.ca:8080/GeneAssociation/