Generating closed frequent gensets under constraints based on FP-Tree structure

C. Trabelsi and C. Latiri†
Research group SOIE
Tunisian High school of management.
24, Avenue de la Libert, 2000 Le Bardo, Tunis, Tunisie
E-mail: Chiraz.Trabelsi@fst.rnu.tn
†Research group URPAH
Faculty of Sciences of Tunis.
Computer Sciences Department
Campus universitaire El Manar, Tunis, Tunisie
E-mail: Chiraz.Latiri@gnet.tn

Abstract - The mechanism of gene regulation is of great interest for biologists, especially in the genomic field. One part of mechanisms controlling the genes expression is provided by the transcription factors, which are proteins that can either repress or stimulate the transcription of a gene. In this paper, we propose a new data mining algorithm, based on boolean contexts, in order to extract a priori relevant frequent closed gensets, i.e., sets of tissue and associated sets of genes and transcription factors which are useful for the biologist. The key feature of our algorithm is a better compromise between the size of the search space and the conveyed discovered knowledge in bioinformatics. For this, the proposed algorithm, called MC²G for Mining Constraint Closed Gensets, uses the Frequent Pattern Tree (FP-Tree) structure, which is an extended Prefix-Tree structure, to prune the search space. Moreover MC²G enables to define statistical and syntaxic constraints on the desired frequent closed gensets and uses them during the extraction process. Experimental comparisons with other algorithms are achieved on real world datasets.

Keywords— Gene expression, Transcription factor, Closed frequent genset, Pattern discovery, Constraint-based data mining, FP-Tree structure, Formal concepts.

I. Context and motivations

DNA Microarray technology provides biologists with the ability to measure the expression of large sets of genes in a single experience. When data from such experiences are stored, it is useful to have accurate means for assigning functions to genes. Also, the interpretation of large-scale gene expression data provides opportunities for developing novel mining methods for selecting for example good drug candidates (all genes are potentially drug targets) from among tens of thousands of expression patterns [GAN 04].

However, an important and challenging question facing biologists is to understand the varied and complex mechanisms that regulate gene expression. One part of mechanisms controlling the genes expression is provided by the transcription factors (TFs), which are proteins that can either repress or stimulate the transcription of a gene. For this purpose, finding the combination of transcription factors for a gene of interest is crucial. It is natural to assume that a target gene and a combination of genes encoding transcription factors for the target should be observed in the same tissues simultaneously.

We formalize combination of transcription factors as an association rule between genes. We consider all tissues (or cells) as baskets, and we treat all genes as items. So, we aim to discover association rules of the form: $\text{gene}_1, \text{gene}_2, \ldots, \text{gene}_k \Rightarrow \text{Target gene}$ where $\text{gene}_1, \text{gene}_2, \ldots$ and $\text{gene}_k$ encode transcription factors. The expression of the target gene and the expression of all genes, $\text{gene}_1, \text{gene}_2, \ldots, \text{gene}_k$, should be correlated.

Hence, in bioinformatics, gene expression data can be represented as boolean matrices (see table I). Columns denoting genes and lines are boolean attributes that enable to record gene and transcription factors occurrences. For instance, in table I, $\text{Tissu}_3$ contains the genes $\text{gene}_1, \text{gene}_2, \text{gene}_3, \text{TF}_1$.

<table>
<thead>
<tr>
<th>$R$</th>
<th>$\text{gene}_1$</th>
<th>$\text{gene}_2$</th>
<th>$\text{gene}_3$</th>
<th>$\text{TF}_1$</th>
<th>$\text{TF}_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{Tissu}_1$</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$\text{Tissu}_2$</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$\text{Tissu}_3$</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$\text{Tissu}_n$</td>
<td>\ldots</td>
<td>\ldots</td>
<td>\ldots</td>
<td>\ldots</td>
<td>\ldots</td>
</tr>
</tbody>
</table>

TABLE I
Example of boolean context for gene expression in various tissues

While considering the data mining context, the frequent gensets mining problem concerns the computa-
tion of sets of genes that are expressed together in enough tissues, i.e., given a frequency threshold called $G_{\text{minsupp}}$. The typical case of basket analysis (huge - eventually millions - number of transactions, hundreds of attributes, but sparse and lowly-correlated data) in data mining has been handled by many algorithms, including the various Apriori-like algorithms that have been designed during the last decade [AGR 94]. However, when the data are dense and highly-correlated, these algorithms fail but the so-called condensed representations of the frequent gensets can be computed. For instance, efficient algorithms can compute the frequent closed sets from which every frequent set and its frequency can be derived without scanning several times the datasets [ZAK 99]. Other important applications concern datasets with only a few transactions, e.g., for typical gene expression data where items denote gene expression properties in biological situations. So, it is possible to use the properties of Galois Connection to compute the closed sets on the smaller dimension and derive the closed sets on the other dimension.

In this work we generate Closed Frequent Gensets (CFG) by efficiently using the FP-tree, the data structure that was first introduced in [PEI 00]. The FP-tree has been shown to be one of the most efficient data structures for mining frequent patterns and for “iceberg” data cube computations [WAN 03], [GRA 03], [HAN 04], [RIO 04]. Thus, we propose a novel algorithm called $\text{MC}^2\text{G}$ that computes closed frequent gensets under constraints. Our contribution is that $\text{MC}^2\text{G}$ algorithm can be used in dense boolean datasets with an active use of constraints. It also enlarges the applicability of concept discovery for matrices whose dimensions are very huge.

The paper is organized as follows. Section 2 presents basic concepts. Section 3 deals with our Mining Frequent Closet Gensets approach and details $\text{MC}^2\text{G}$ algorithm. Section 4 provides an experimental evaluation. Finally, we present our work in progress.

II. Basic Concepts

Let $G = g_1, g_2, \ldots, g_n$ be a set of genes. A genset $X$ is a non-empty subset of $G$. A boolean context $E$ is defined as a triplet $< T, G, R >$ where $T$ is a set of tissus, $G$ is a set of genes and $R$ is a binary relation. Each biological tissu $T_x$ is a pair $< \text{tid}, X >$, where $\text{tid}$ is the tissu identifier, and $X$ is a genset. A tissu $T_x = < \text{tid}, X >$ is said to contain a genset $Y$, if $Y \subseteq X$. The support (i.e., frequency value) of a genset $X$, denoted as $\text{supp}(X)$, is the number of tissus in $E$ which contain $X$.

To introduce our approach, we consider the following definitions:

**Definition 1:** (Frequent genset) A genset is called a frequent genset if its support is greater than the minimal support threshold $G_{\text{minsupp}}$. Given a threshold $G_{\text{minsupp}}$, the set $L$ of all the frequent gensets in $E$ is defined as $L = \{ X \subseteq G | \text{supp}(X) \geq G_{\text{minsupp}} \}$.

**Definition 2:** (Maximal frequent genset) A frequent genset is called a maximal frequent genset if there is no other frequent genset to be its proper superset.

**Definition 3:** (Frequent closed genset) A genset $X$ is called a closed genset if genset $Z$ such that: (i) $Z$ is a proper superset of $X$, and (ii) $\text{supp}(Z) = \text{supp}(X)$. Note that if either condition (i) or (ii) does not hold for $Z$, then $X$ can be a closed genset. So, a closed genset $X$ is a frequent closed genset (FCG) if it is frequent.

From the above definitions, we can derive the following lemmas:

**Lemma 1:** Let $X$ and $Y$ be two gensets. If $X$ is a proper superset of $Y$, then $\text{supp}(X) = \text{supp}(Y)$.

**Lemma 2:** Let $X$ and $Y$ be two gensets with the same support. $Y$ is not a closed genset if $Y$ is a proper subset of $X$. (This can be derived directly from Definition 3.)

**Lemma 3:** A maximal frequent genset is a frequent closed genset. (This can be derived directly from Definition 3.)

While considering the above definitions and lemmas, we have a more concise definition of the frequent closed genset.

**Definition 4:** (Frequent Closed Genset FCG) A frequent closed genset is a maximal frequent genset, or a frequent genset whose support is higher than the supports of all its proper supersets.

In our approach we consider two types of constraint:

- statistical constraint dealing with the support of the frequent closed gensets,
- syntactic constraint which prunes the space research of frequent closed gensets by considering only the factors transcription while deriving the FP-tree.

So, we give a more formal definition:

**Definition 5:** (Monotonic and anti-monotonic constraints) Given $L$ a collection of sets, a constraint $C$ is said anti-monotonic if and only if $\forall a, b \in L$ such that $a \subseteq b$, $C(a) \Rightarrow C(b)$. $C$ is said monotonic if and only if $\forall a, b \in L$ such that $a \subseteq b$, $C(a) \Rightarrow C(b)$.

For example, in A-priori like algorithms [AGR 94], the minimal frequency constraint (on $L_G$) is used to prune the search space. This constraint is anti-monotonic on $L_T$ and can be considered as monotonic on $L_G$ because when a set of genes is larger, the associated set of tissus is smaller.

However, finding all closed frequent gensets is the basic step of association rule mining process since the non-redundant association rules can be inferred from all the closed frequent gensets. In the next section and with respect to the above definitions, we present $\text{MC}^2\text{G}$ for discovering the frequent closed gensets under the cited constraints and generat-
ing association rules between transcription factors and genes.

III. Mining Frequent Closed Gensets

A. The FP-tree structure

The FP-tree (Frequent Pattern tree) is a compact representation of all relevant frequency information in a database [GRA 03]. In our approach, every branch of the FP-tree represents a frequent genset, and the nodes along the branches are stored in decreasing order of the corresponding genes frequency, with leaves representing the least frequent genes. Compression is achieved by building the tree in such a way that overlapping gensets share prefixes of the corresponding branches. The FP-tree has a header table associated with it. Single genes and their counts are stored in the header table in decreasing order of their frequency. The entry for a gene also contains the head of a list that links all the corresponding nodes of the FP-tree. Compared with Apriori algorithm [AGR 94] and its variants which need several database scans, the use of FP-tree structure only needs two dataset scans to mining all frequent gensets. The first scan counts the number of occurrences of each gene. The second scan constructs the initial FP-tree which contains all frequency information of the original dataset. Hence, to construct the FP-tree, we extract first all frequent genes by an initial scan of the database. Then, we insert these genes in the header table, in decreasing order of their count. In the next (and last) scan, the set of frequent genes in each scanned tissu are inserted into the FP-tree as a branch. If a genset shares a prefix with a genset already in the tree, the new genset will share a prefix of the branch representing this genset. In addition, a counter is associated with each node in the tree which stores the number of transactions containing the genset represented by the path from the root to the focused node. This counter is updated during the second scan, when a tissu causes the insertion of a new branch.

\[ \text{In figure 1, (a) shows an example of a dataset and (b) depicts the FP-tree for that dataset. Note that there may be more than one node corresponding to a gene in the FP-tree. The frequency of any one gene } g_i \text{ is the sum of the count associated with all nodes representing } g_i, \text{ and the frequency of a genset is equal to the sum of the counts of the least frequent gene in it, restricted to those branches that contain the genset. For instance, from figure 1 (b) we can see that the frequency of the genset } g_2g_1 \text{ is 2. Thus the constructed FP-tree contains all frequency information of the dataset. From the recent publications, we cite the FP-growth method by Han and al [HAN 04] which also uses the FP-tree data structure and based on the following principle: if } G_1 \text{ and } G_2 \text{ are two gensets, the count of genset } G_1 \cup G_2 \text{ in the database is the same one of } G_2, \text{ in the restriction of the database to those tissus containing } G_1. \text{ This restriction of the database is called the conditional pattern base or conditional context of } G_1, \text{ which we denote by } CC_{G1}, \text{ and the FP-tree constructed from the conditional pattern base is called } G_1 \text{ conditional FP-tree. The above procedure is applied recursively, and it stops when the resulting new FP-tree contains only one single path. The complete set of frequent gensets is generated from all single-path FP-trees.}

B. Mc^2G Approach

The main work done in the FP-growth method is traversing FP-tree and constructing new conditional FP-trees after the first one derived from the original database. Several experimental results have shown that about 80% of the CPU time was used for traversing FP-trees [GRA 03]. However, in our approach, we propose to built a complete FP-tree with a reduced dataset. Since, only tissus containing transcription factors will be considered. According to this, the size of the tree is usually much smaller than its original dataset. Thus, our approach holds on three steps:

Step 1: Boolean context construction:
1) Scan the dataset. Find the support of each gene and transcription factor.
2) Construct the boolean context } \hat{E} \text{ for frequent genes and frequent transcription factors according to the minimum support threshold.
3) Sort } \hat{E} \text{ in support-descending order.

Step 2: FP-tree construction
1) Collect } TF, \text{ the set of frequent transcription factors, and the support of each frequent transcription factor.
2) Sort } TFs \text{ in support-descending order in } TF-List, \text{ the list of frequent transcription factors.
3) Create the root of an FP-tree, } Tr, \text{ and label it...}
as “null”. For each tissue \( t_i \) in \( \tilde{E} \) do the following, i) Select the frequent genes in \( t_i \). ii) Let the sorted frequent-gene list in \( t_i \) be \([h|r]\), where \( h \) is the first element and \( r \) is the remaining list. Call the function \text{INSERT-TREE}([h|r], Tr) which is performed as follows: If \( Tr \) has a child \( N \) such that \( N.item - \text{name} = r.item - \text{name} \), then increment \( N \)'s count by 1; else create a new node \( N \), with its count initialized to 1, its parent is linked to \( Tr \), and its node-link is linked to the nodes with the same item-name via the node-link structure. If \( r \) is nonempty, call \text{INSERT-TREE}(r, N) recursively. The FP-tree construction needs two scans of the dataset: The first scan collects the set of frequent genes in \( t \). If \( E \) contains the subset of tissues in the boolean context \( \tilde{E} \) containing \( g_i \), and all the occurrences of local infrequent genes \( g_i \), and genes following \( g_i \) in the local genes list sorted in support descending order are omitted. 2) If a genset \( G_1 \) is the maximal set of genes appearing in every tissue in \( G_2 \)-conditional context, and \( G_1 \cup 2 \) is not subsumed by some already frequent closed genset with identical support, then add \( G_1 \cup G_2 \) to \( EFGC \). The set of frequent closed gensets. In the rest, we give the pseudo-code of \( \text{MC}^2 G \) Algorithm:

C. \( \text{MC}^2 G \) Algorithm

\begin{algorithm}
\caption{\( \text{MC}^2 G \) Algorithm to extract frequent closed gensets under constraints}
\textbf{Input:} \( E = \langle T,G, R \rangle \): Boolean context and \( G_{\text{min supp}} \) threshold
\textbf{Output:} \( EFGC \): The set of frequent closed gensets
1) \( EFGC = \emptyset \)
2) Pruning the boolean context \( E = \langle T,G,R \rangle \) relating to \( G_{\text{min supp}} \) value */
3) Sorting \( E = \langle T,G,R \rangle \) and constructing \( TF\)-list which is the transcription factors list satisfying \( G_{\text{min supp}} \) and sorted in support descending order */
4) \( \hat{E} = \text{SORT} (E) \)
5) \( TF\)-list = \{\( TF : g_i \) where \( TF : g_i \) is a frequent 1-genset \}
6) Construction of the FP-tree */
7) for all \( TF : g_i \in TF\)-list do
8) \( EFGC_i = \text{COMPUTE-CLOSURE} (TF : g_i, \hat{E}) \)
9) \( EFGC = EFGC \cup EFGC_i \)
10) Return \( EFGC \)
\end{algorithm}

Property 1: A closed genset \( G \) extracted from a conditional context is formed by the concatenation of the 1-gensets which have the same support as \( G \) in the same conditional context.

Property 2: If a genset \( G \) extracted from a conditional context can’t be concatenated with the 1-gensets which have the same support as \( G \) in the same conditional context then \( G \) is a closed genset.

Property 3: It is useless to develop a conditional context of a genset \( G \) which is contained in a closed frequent genset \( G' \) already extracted such that \( \text{supp}(G) = \text{supp}(G') \).

Our algorithm \( \text{MC}^2 G \) considers the transcription factors list \( TF\)-list one by one beginning by the less frequent one. \( \text{MC}^2 G \) construct then for every transcription factor \( TF : G_i \) of \( TF\)-list, the corresponding FP-tree conditional which contains the set of tissues where \( TF : G_i \) is expressed, represented by 1 in the boolean context. In this FP-tree, the transcription factor \( TF : G_i \) is omitted with all those already considered.

\begin{algorithm}
\caption{\text{COMPUTE-CLOSURE}}
\textbf{Input:} \( EFGC \): The set of frequent closed gensets \( G \)
\textbf{Output:} \( EFGC \): The set of frequent closed gensets \( G \)
1) \( EFGC = \emptyset \)
2) *Step 1: Create the root node */
3) *Step 2: Construct the conditional context \( CC \) */
4) \( CC = \text{CONSTRUCT-CC} (G, EFGC) \)
5) *Step 3: Construct the FP-tree */
6) for all \( G_k \in CC \) do
7) if \( \text{supp} (G_k) = \text{supp} (G) \) then
8) if \( G_k \in EFGC \) then
9) \( \text{INSERT-NODE-FP-TREE}(G_k, EFGC) \)
10) \( EFGC = EFGC \cup G_k \)
else
11) \( \text{INSERT-NODE-FP-TREE}(G_k) \)
12) \( \text{COMPUTE-CLOSURE}(G_k, CC_k) \)
13) Return \( EFGC \)
\end{algorithm}

So, recursively and proceeding by depth-first search, our algorithm extracts all closed frequent gensets containing necessarily transcription factors, regarding to the syntactic constraint. However, to prune the search space and to reduce the cost of the extraction process, \( \text{MC}^2 G \) algorithm applies the property 3 since the construction of the conditional FP-tree of a genset \( G_i \) is performed only for the \( G_i \) which are not covered by a frequent closed genset already extracted. The pseudo-code of \( \text{MC}^2 G \) is given in the algorithm 1.

IV. Experimental evaluation

We compare the number of potential frequent closed gensets, i.e., \( FCG \), extracted with \( \text{MC}^2 G \) algorithm with those given using \( \text{FP-CLOSE} \) algorithm [GRA 03] and \( \text{D-MINER} \) algorithm [BES 04]. The three algorithms were tested on two types of gene expres-
Procedure **CONSTRUCT-CC**

**Input:** A frequent $G = k$-genset and $CE$ a sorted boolean context

**Output:** $CC_G$: the conditional context related to $G$ which is a list of genes expressed in the same tissue of $CE$ with their respective supports

$CC_G = \emptyset$

/* Scanning tissue $t_i \in CE /*$

for all $t_i \in CE$
do

if $G \in t_i$ then

/* Scanning $g_k \in t_i /*$

for all $(g_k \in t_i)$ and $(g_k \neq G)$ do

$g_k$.genset = $g_k$

$g_k$.supp = supp($g_k$)

$CC_G.supp = supp(g_k)$

$CC_G = CC_G \cup g_k$

Return ($CC_G, supp(CC_G)$)

Evaluation datasets provided by Pasteur Institut of Tunis [GIT 02], [REY 02]. The Table II shows their characteristics. FP-Close is a column enumeration-based frequent closed pattern mining algorithm, it performs a depth-first search of closed frequent itemsets using the FP-tree structure. D-Miner also computes closest frequent itemsets (or formal concepts) but it works differently from FP-Close algorithm since it doesn’t use a special data structure to prune the search space but an active use of constraints. $MC^2G$ aims to be a better trade-off between FP-Close and D-Miner algorithms in the uses of both constraints and FP-Tree structure to reduce the search space and to optimize the extraction of the frequent closed gensets from the datasets.

Our experimental results show that even search space and frequent closed gensets number are reduced while considering constraints. For each data, we vary the value of the support as the percentage of the number of total FCG extracted.

The results are depicted in Figure 1 and Figure 2. We notice the following observations: First, we observe an important variation in the number of the FCG extracted for the three algorithms. As expected, the number of FCG decreases when the value of $G_{\text{minsupp}}$ increases, because a larger value of $G_{\text{minsupp}}$ enables representative pattern to cover more patterns. Increasing the value of $G_{\text{minsupp}}$ also shifts the distributions of the FCG to lower support. We also remark that $MC^2G$ outperforms D-Miner which in form outperforms FP-Close. This seems quite natural since FP-Close does not consider constraints during the mining process. For example, for the 2 $G_{\text{minsupp}}$ value, the number of FCG extracted with $MC^2G$ is 36% less than those of FP-Close and 22% less than those given by D-Miner algorithm. Second, on both sparse and dense datasets, the overall shape of the distributions of the extracted FCG of the three algorithms are similar. Obviously the number of the FCG extracted by performing $MC^2G$ substantially decreases as the syntactical constraint become more selective. This is because $MC^2G$ extract only the frequent closed gensets containing transcription factors, so the search space will be considerably reduced. As we can see, from our experimental results, $MC^2G$ outperforms FP-CLOSE and D-MINER algorithms. This means that $MC^2G$ is quite effective independently of the dataset.

### Table II: Datasets characteristics

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Nature</th>
<th>#tissus</th>
<th>#genes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mining1</td>
<td>sparse</td>
<td>12</td>
<td>148</td>
</tr>
<tr>
<td>Mining2</td>
<td>dense</td>
<td>16</td>
<td>171</td>
</tr>
</tbody>
</table>

Fig. 2. Variation curves related to the dataset Mining 1

Fig. 3. Variation curves related to the dataset Mining 2

V. Work in progress: Mining association rules between genes

Most of the data mining methods adapted for gene expression data analysis are based on the classifica-
tion or the clustering techniques of data mining. Little work has been done using association rule mining. These methods (supervised and unsupervised) are useful for grouping and identify genes of unknown functions by their resembling characteristics to other genes of known functions [JEU 05], [RIO 04], [SOU 05], [PAN 03]. While the existing tools are useful for determining membership of genes by similarity, they do not identify the regulatory relationships among genes that are found in the same class of molecular actions. For example, when using clustering tools, we can say that genes A, B and C are closely related in their expression pattern. But we cannot say anything about the relationship among A, B, and C. With association rule mining, we can take a step further and may be able to discover relationships such as “A ⇒ B, C”, that when gene A is expressed, B and C are also expressed with a measure of confidence c.

With our Mc²G algorithm, we generate directly from the frequent closed gensets association rules between transcription factors and genes. An association rule is a strong one when its support (i.e., frequency of the represented pattern) and confidence (i.e., strength of the dependency between the premise and the conclusion) are higher than minimum thresholds fixed by the user (minsupp and minconf). However, the number of association rules generated is often huge because many of them are redundant. So, we propose in our approach restrict rules extraction to a reduced set containing only non-redundant ones which are strictly related to the biologist need [LAT 05].

The procedure Bio-gen-rules and the experimental evaluation will be presented in a forthcoming paper.

VI. Conclusion

Computing closed frequent itemset has been proved useful in many application domains but remains extremely hard from dense boolean data sets. We have described a new algorithm that computes closed frequent gensets under two categories of constraints. First, it can be used for closed gensets computation and thus for association rules discovery. Next, for difficult contexts, i.e., dense boolean matrices where the dimension is large, the analyst can provide monotonic constraints on desired gensets. We are now working on the biological validation of the extracted gensets. We also have to experiment other kind of constraints and compare the number of association rules derived with other algorithms. We believe that association rule mining can be used as an exploratory tool for identifying interesting possible gene functions, relationships, and biological pathways that can be further studied by designing experiments targeting these genes.

REFERENCES


[HAN 04] Han J., Pei J., Yin Y., Mao R., Mining Frequent Patterns without Candidate Generation: A frequent-pattern tree approach, Data Mining and Knowledge Discovery, n8, p. 53-87, 2004.


