On the gene team mining problem

Hao-Sen Chen, Guanling Lee, Sheng-Lung Peng*
Department of Computer Science and Information Engineering
National Dong Hwa University, Hualien 974, Taiwan
*Corresponding author: slpeng@mail.ndhu.edu.tw

Abstract—Let Σ be a set of n genes. A chromosome G can be represented as a permutation of Σ. A subset D of Σ is a δ-set of G if two consecutive genes in G∩D have distance at most δ. For a set G of m chromosomes, a set D is a δ-team of G if D is a δ-set of every chromosome of G. Given a gene set Σ, a chromosome set G, and an integer δ, the gene team mining problem is to find all possible δ-teams of G. Given a gene set Σ, a chromosome set G of m chromosomes, an integer k ≤ m, and an integer δ, the gene team mining problem is to find all possible δ-teams for any possible chromosome set G′ such that G′ ⊆ G and |G′| ≥ k.

In this paper, we study the gene team mining problem. It is known that the Apriori technique is used wildly in data mining. However, the gene team mining problem has no Apriori property that all nonempty subsets of a δ-team (δ-set) must also be a δ-team (δ-set). Thus, many techniques used in data mining cannot be applied for this gene team mining problem. In this paper, we propose a concept of pseudo-support. By using this concept, an Apriori-like algorithm can be obtained to solve the gene team mining problem.

I. INTRODUCTION

It is known that a gene is a contiguous stretch of DNA that contains the information necessary to build a protein. Genes are stored in a very long DNA molecule, called chromosome. Biological evidence suggests that genes which are located close tend to code for proteins that have a functional interaction [1]. Later, Luc et al. proposed the concept of gene teams [2]. Let Σ be a set of n genes, a chromosome G can be represented as a permutation of Σ, where each gene g in G associates an integer to denote the location of g in G. For a given value δ, a subset of Σ is called a δ-term of a chromosome if the distance between two neighbor genes is smaller or equal to δ. For example, let Σ = {a, b, c, d}. Given a set of chromosomes $G = \{G_1, G_2\}$, where $G_1 = \langle a_1, b_2, c_3, d_4 \rangle$ and $G_2 = \langle a_1, b_4, c_5, d_6 \rangle$, let δ = 2. Then the possible δ-terms of $G_1$ and $G_2$ are shown in Table I.

<table>
<thead>
<tr>
<th>δ-terms of chromosomes</th>
<th>δ-terms</th>
</tr>
</thead>
<tbody>
<tr>
<td>$G_1 = \langle a_1, b_2, c_3, d_4 \rangle$</td>
<td>${a, b, c, d}$</td>
</tr>
<tr>
<td>$G_2 = \langle a_1, b_4, c_5, d_6 \rangle$</td>
<td>${a}, {b, c}, {d}$</td>
</tr>
</tbody>
</table>

For a given chromosome set G, a δ-term x is called a gene team of G if x is a δ-term in every chromosome of G. A gene team x is non-trivial if |x| ≥ 2. For example, $\{b, c\}$ is a non-trivial gene team of G in the above example while $\{a\}$ and $\{d\}$ are trivial gene teams. The gene team finding problem is to find all possible gene teams that appear in every chromosome. The input is a set G of chromosomes and an integer δ. The output is a set of gene teams.

Consider Fig. 1 as an example. The chromosome set is $G = \{G_1, G_2\}$, where $G_1 = \{a_1, b_2, c_5, d_7, e_8\}$ and $G_2 = \{e_1, d_3, a_7, c_8, b_9\}$. If δ = 3, then $\{a, b, c\}$ and $\{e, d\}$ are non-trivial gene teams.

![Fig. 1. Example for gene team finding problem.](image)

Let |G| = m and |Σ| = n. Béal et al. proposed an $O(mn\lg^2 n)$ algorithm for the gene team finding problem [3]. Zhang and Leong proposed the concept of gene team tree such that gene teams can be represented in the tree for all possible values δ [4]. By applying the technique proposed in [3], the gene team tree can be constructed in $O(mn\lg^2 n)$ time [4]. Recently, Wang and Liu improved the algorithm of Béal et al. to $O(n\lg t)$ time, where t ≤ n is the number of gene teams [5]. By a careful implementation, the algorithm of Zhang and Leong can also be improved to $O(n\lg n\lg\lg n)$ time [5]. He and Goldwasser extended the model such that duplicated genes are allowed [6]. For the case of m = 2, they proposed an $O(n_1n_2\lg t)$ algorithm for this problem, where $n_1$ and $n_2$ are the lengths of the two input chromosomes, respectively. Other related works to the gene team finding problem, please refer to [7]–[10].

The gene team mining problem is to find all possible δ-terms that appear in at least k genomes. For this problem, we need a δ value and an additional value called the minimum support (min_support) that is a value between 0 and 1. For a δ-term x, let support(x) be defined as follows.

$$\text{support}(x) = \frac{\# \text{ of genomes that contain } x}{\text{number of all genomes}}$$

In the gene team mining problem, we call x a frequent gene team of G if support(x) ≥ min_support.

Consider Fig. 2 as an example. Assume that the genome set is $G = \{G_1, G_2, G_3\}$, where $G_1 = \{a_1, b_2, c_5, d_7, e_8\}$, $G_2 = \{e_1, d_3, a_7, c_8, b_9\}$, and $G_3 = \{a_1, b_4, e_4, d_7, c_9\}$. If δ = 3 and min_support is 50%, then $\{d, e\}$ is a frequent gene team of G since support($\{d, e\}$) = 100% ≥ min_support. However
\{a, d\} is not a frequent gene team since \text{support}(\{a, d\}) = 0. Similarly, \{c, d\} is also a frequent gene team of \mathcal{G} since \text{support}(\{c, d\}) = 66.7\%.

Similarly, \{c, d\} is also a frequent gene team of \mathcal{G} since \text{support}(\{c, d\}) = 66.7\%.

Fig. 2. Example for gene team mining problem.

Gene team tree provides a compact representation of gene teams for all gap lengths. Thus, it can be used to mine interested teams for a set of chromosomes. However, in data mining research, we assume that the size of input data is larger than the memory. That is, we cannot keep the whole data in the memory for processing. Therefore, the gene team tree is no longer suitable for the gene team mining problem. The 

A. Apriori property

The Apriori algorithm [11] is well known for solving many data mining problems. The purpose of Apriori algorithm is for mining frequent patterns in a database. It contains two main phases. The first one is a candidate generation rule. The second one is a candidate checking rule. It works from bottom to top. The key point for the Apriori algorithm is the Apriori property. For example, if the set \{a, b, c, d\} is frequent, then its all nonempty subsets are also frequent. Fig. 4 shows an example of Apriori Property.

After deleting all the non-frequent itemsets, the remaining candidate sets are the desired answer (see Fig. 6). However, the Apriori property cannot be applied in the gene team mining problem. The following example shows the case.

\begin{itemize}
\item \(\{a, b, c, d\}\) is not frequent if \(\text{support}(\{a, b, c, d\}) = 0\).
\item \(\{a, c, d\}\) is frequent if \(\text{support}(\{a, c, d\}) = 66.7\%\).
\end{itemize}

Fig. 4. Apriori property (1/3).

In this example, if the set \(\{a, d\}\) is not frequent, then \(\{a, b, d\}\) and set \(\{a, c, d\}\) must also not be frequent. Similarly, once the sets \(\{a, b, d\}\) and \(\{a, c, d\}\) are not frequent, then \(\{a, b, c, d\}\) will not be frequent. This concept is called Apriori property (see Fig. 5).

After deleting all the non-frequent itemsets, the remaining candidate sets are the desired answer (see Fig. 6). However, the Apriori property cannot be applied in the gene team mining problem. The following example shows the case.

Fig. 5. Apriori property (2/3).

To improve the efficiency of the level-wise generation of frequent sets, an important property called the Apriori property, i.e., all nonempty subsets of a frequent set must also be frequent. This paper considers the problem of mining the frequent gene teams in \(m\) genomes with the given \text{min\_support} value and distance \(\delta\) by using an extended Apriori algorithm. Our extended Apriori algorithm uses the concept of additional \text{pseudo\_support} to mine all possible gene teams, where some teams cannot be found by traditional Apriori algorithm. The remaining organization of this paper is as follows. Section 2 presents our algorithm for the gene team mining problem. Section 3 shows the simulation on some random data using our algorithm. Finally, we give a conclusion in Section 4.

II. OUR ALGORITHM

Recall that in the gene team mining problem, we need a \(\delta\) value and an additional value \text{min\_support} called the minimum support. The gene team mining problem is to find all possible \(\delta\)-terms that appear in at least \(k\) genomes. The input are a set of genomes, minimum support \(k\), and an integer \(\delta\).

A. Apriori property

The Apriori algorithm [11] is well known for solving many data mining problems. The purpose of Apriori algorithm is for mining frequent patterns in a database. It contains two main phases. The first one is a candidate generation rule. The second one is a candidate checking rule. It works from bottom to top. The key point for the Apriori algorithm is the Apriori property. For example, if the set \(\{a, b, c, d\}\) is frequent, then its all nonempty subsets are also frequent. Fig. 4 shows an example of Apriori Property.

In this example, if the set \(\{a, d\}\) is not frequent, then \(\{a, b, d\}\) and set \(\{a, c, d\}\) must also not be frequent. Similarly, once the sets \(\{a, b, d\}\) and \(\{a, c, d\}\) are not frequent, then \(\{a, b, c, d\}\) will not be frequent. This concept is called Apriori property (see Fig. 5).

After deleting all the non-frequent itemsets, the remaining candidate sets are the desired answer (see Fig. 6). However, the Apriori property cannot be applied in the gene team mining problem. The following example shows the case.
Let $\mathcal{G} = \{G_1, G_2\}$, $G_1 = \{a_2, b_5, c_7, d_{10}\}$, $G_2 = \{d_1, c_4, b_5, a_8\}$, $\delta = 3$, and $\text{min}_\text{support} = 100\%$. Then the answer is $\{a, b\}$, $\{b, c\}$, $\{c, d\}$ by Apriori property. However, the exact answer should include the sets $\{a, b, c, d\}$, $\{a, b, c\}$, and $\{b, c, d\}$. It is easy to check that the answer does not admit the Apriori property. The set $\{b, c, d\}$ does not appear in the answer of Apriori algorithm because it does not have the Apriori property. Thus the traditional Apriori algorithm cannot work for the gene team mining problem.

**B. Extended Apriori algorithm**

Our idea is to add an extra support, called pseudo support. Consider the last example again. We can see that the support of $\{a, b\}$ is 2. So it is not a problem. However consider the set $\{a, c\}$. Though $a$ and $c$ are too far away, they belong to the same $\delta$-term in each genome. In this case, we count its pseudo support as 2 though its support is 0. Thus for each set, we define the total support as the sum of its support and its pseudo support. Therefore, the set $\{a, c\}$ can be a candidate 2-set for generating 3-sets. That is, by using an extra variable pseudo support, we can correctly generate all possible $\delta$-terms. Then by a checking we can obtain all possible gene teams. Algorithm 1 shows our main algorithm.

Algorithm 2 is the procedure for candidate generation. In the algorithm, for two gene sets $c_1$ and $c_2$, if $c_1[1] = c_2[1], \ldots, c_1[r-2] = c_2[r-2]$ and $c_1[r-1] < c_2[r-1]$, where $r-1$ is the size of $c_1$, then the join of $c_1$ and $c_2$ are defined as $c_1 \Join c_2 = \{c_1[1], c_1[2], \ldots, c_1[r-2], c_1[r-1], c_2[r-1]\}$. Note that the notation $\Join$ is borrowed from [11].

Our main idea of pseudo support is in the Procedure Check. The gene set $c$ remains in the candidate set if its support plus its pseudo support is greater than or equal to the min support. The detail is in the Algorithm 3.

**III. SIMULATION**

In this paper, we implement our extended Apriori algorithm for the gene team mining problem. Our environment uses machine of CPU: P8700 2.53GHz, RAM: DDR-2 4GB, OS: Windows 7.

**A. Simulation 1.**

In the first test data set, the number of genes is 1000. There are three data in this test data set, with numbers of genomes (chromosomes) being 2500, 5000, and 7500, respectively. The min support’s used are 30%, 50%, and 70%. These data are randomly generated with maximum gap 10 between two consecutive genes. In the program, we assume that $\delta = 7$. Fig. 7 shows the results.

In the same simulation, the numbers of gene teams that we mined are shown in Table II. The result shows that our algorithm runs well even if we have to consider 7500 species.

**B. Simulation 2.**

For the next test data set, we consider the different numbers of genes, i.e., 500, 1000, 1500, and 2000. The number of genomes (chromosomes) is 5000. These data are randomly generated with maximum gap 10 for two consecutive genes. In the program, we assume that $\delta = 7$. The min support’s we considered are 30%, 50%, 70%. Fig. 8 shows the results of
Algorithm 3 Check

Input: a gene set $c$ and a distance $\delta$.

Output: true or false.

1: $c.support \leftarrow 0$
2: $c.pseudo_sup \leftarrow 0$
3: $c.total_sup \leftarrow 0$
4: for each $d \in G$ do
5: if all genes in $c$ have the same group no then
6: if $c$ is a $\delta$-team in $d$ then
7: $c.support++$
8: else
9: $c.pseudo_sup++$
10: end if
11: end if
12: end for
13: if $(c.support + c.pseudo_sup)/|G| \geq \text{min\_support}$ then
14: return true
15: else
16: return false
17: end if

support, we can extend the traditional Apriori algorithm for the gene team mining problem. Our simulation shows that our algorithm runs in a reasonable time on data of 7500 genomes with 1000 genes or on data of 5000 genomes with 2000 genes. In summary, our algorithm for the gene team mining problem is promotive and practical.

### REFERENCES


