CARPENTER: Finding Closed Patterns in Long Biological Datasets

Feng Pan, Gao Cong, Anthony K. H. Tung *†
Natl. University of Singapore
{panfeng,conggao,atung}@comp.nus.edu.sg

Jiong Yang
University of Illinois, Urbana Champaign
jioyang@cs.uiuc.edu

Mohammed J. Zaki ‡
Rensselaer Polytechnic Institute
zaki@cs.rpi.edu

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications-Data Mining

Keywords

frequent pattern, closed pattern, row enumeration

ABSTRACT

The growth of bioinformatics has resulted in datasets with new characteristics. These datasets typically contain a large number of columns and a small number of rows. For example, many gene expression datasets may contain 10,000-100,000 columns but only 100-1000 rows.

Such datasets pose a great challenge for existing (closed) frequent pattern discovery algorithms, since they have an exponential dependence on the average row length. In this paper, we describe a new algorithm called CARPENTER that is specially designed to handle datasets having a large number of attributes and relatively small number of rows. Several experiments on real bioinformatics datasets show that CARPENTER is orders of magnitude better than previous closed pattern mining algorithms like CLOSET and CHARM.

1. INTRODUCTION

The growth of bioinformatics has resulted in datasets with new characteristics. These datasets typically contain a large number of columns and a small number of rows. For example, many gene expression datasets may contain 10,000-100,000 columns or items but usually have only 100-1000 rows or transactions. Such datasets pose a great challenge for existing frequent pattern discovery algorithms. While there are a large number of algorithms that had been developed for frequent pattern mining [1, 5, 11], their running time increases exponentially with increasing average row length, thus such high-dimensional data renders most current algorithms impractical. This also holds for extant methods for mining closed patterns [6, 2, 7, 10].

Previous (closed) frequent pattern mining methods work well for datasets with small average row length, since if i is the maximum row size, there could be 2^i potential frequent itemsets; usually $i < 100$. However, for the gene expresion datasets in bioinformatics domain i can be in the range of tens of thousands. As a result, search over the itemset space is impractical. Since these datasets have a small number of rows (say m) usually in the range of a hundred or a thousand (i.e., $m \ll i$), it appears reasonable to design an algorithm that searches the row set space instead of the usual itemset space.

In this paper, we describe a new algorithm called CARPENTER [1], that is specially designed to handle datasets having a large number of items and relatively small number of rows. CARPENTER is a novel algorithm which discovers frequent closed patterns by performing depth-first row-wise enumeration instead of the usual itemset enumeration, combined with efficient search pruning techniques, to yield a highly optimized algorithm. Our experiments show that this unconventional approach produces good results when mining long biological datasets and outperforms current methods like CHARM[10] and CLOSET[7] by more than an order of magnitude.

2. PRELIMINARIES

Let $F = \{f_1, f_2, .., f_m\}$ be a set of items, also called *features*. Our dataset D consists of a set of rows $R = \{r_1, .., r_n\}$, where each row r_i is a set of features, i.e., $r_i \subseteq F$. Figure 1(a) shows an example dataset in which the features are represented using alphabets a through t. There are altogether 5 rows, r_1,...,r_5. The first row r_1 contains the feature set $\{a, b, c, l, o, s\}$. For convenience, in the sequel, we drop set notation and denote a set of features $\{a, c, f\}$ as acf, and we denote a row set $\{r_2, r_3, r_5\}$ as 235.

Given a set of features $F' \subseteq F$, we define the **feature support set**, denoted $\mathcal{R}(F') \subset R$, as the maximal set of rows that contain F'. Likewise, given a set of rows $R' \subset R$, we define the **row support set**, denoted $\mathcal{F}(R') \subset F$, as the maximal set of features common to all the rows in R'.

*Contact Author

†This work was supported in part by NUS ARF grant R252-000-121-112 and R252-000-142-112.

‡This work was supported in part by NSF CAREER Award IIS-0092978, DOE Career Award DE-FG02-02ER25538, and NSF grant EIA-0103708.

SIGKDD '03, August 24-27, 2003, Washington, DC, USA

[1]CARPENTER stands for Closed Pattern Discovery by Transposing Tables that are Extremely Long; the "ar" in the name is gratuitous.

i	r_i
1	a,b,c,l,o,s
2	a,d,e,h,p,l,r
3	a,c,e,h,o,q,t
4	a,e,f,h,p,r
5	b,d,f,g,l,q,s,t

(a) Example Table

f_j	$\mathcal{R}(f_j)$
a	1,2,3,4
b	1,5
c	1,3
d	2,5
e	2,3,4
f	4,5
g	5
h	2,3,4
l	1,2,5
o	1,3
p	2,4
q	3,5
r	2,4
s	1,5
t	3,5

(b) Transposed Table, TT

Figure 1: Running Example

f_j	$\mathcal{R}(f_j)$
a	4
e	4
h	4

Figure 2: $TT|_{\{2,3\}}$

As an exmaple consider Figure 1(a). Let $F' = aeh$, then $\mathcal{R}(F') = 234$ since these are all the rows that contain F'. Also let $R' = 23$, then $\mathcal{F}(R') = aeh$ since it is the maximal set of features common to both r_2 and r_3.

Given a set of features F', the number of rows in the dataset that contain F' is called the **support** of F'. By definition, the support of F' is given as $|\mathcal{R}(F')|$. A set of features $F' \subseteq F$ is called a **closed pattern** if there exists no F'' such that $F' \subset F''$ and $|\mathcal{R}(F'')| = |\mathcal{R}(F')|$, i.e., there is no superset of F' with the same support. Put another way, the row set that contains superset F'' must not be exactly the same as the row set of F'. A feature set F' is called a *frequent closed pattern*, if it is i) closed, ii) $|\mathcal{R}(F')| \geq minsup$, where $minsup$ is a user specified lower support threshold. For example, given $minsup = 2$, the feature set aeh is a frequent closed pattern in Figure 1(a) since it occurs three times. ae, on the other hand, is not a frequent closed pattern, since it is not closed ($|\mathcal{R}(aeh)| = |\mathcal{R}(ae)|$), although its support is more than $minsup$.

Given a dataset D which contains records that are subset of a set of features F, our problem is to discover all frequent closed patterns with respect to a user support threshold $minsup$. In addition we assume that the dataset satisfies the condition $|R| \ll |F|$.

3. THE CARPENTER ALGORITHM

To illustrate CARPENTER , we will use the tables in Figure 1 as a running example. Table 1(b) is a transposed version of Table 1(a), denoted TT. In TT, each tuple lists a feature, along with the row ids where that feature occurs in the original table. As an example 15 is the set of rows that contain feature b, which produces the second tuple in TT. In the sequel we always refer to entries of the transposed table as tuples, and entries of the original table as rows.

Unlike existing algorithms which perform their search by enumeration of feature sets [6, 7], CARPENTER performs search by enumeration of row sets. Figure 3 illustrates the

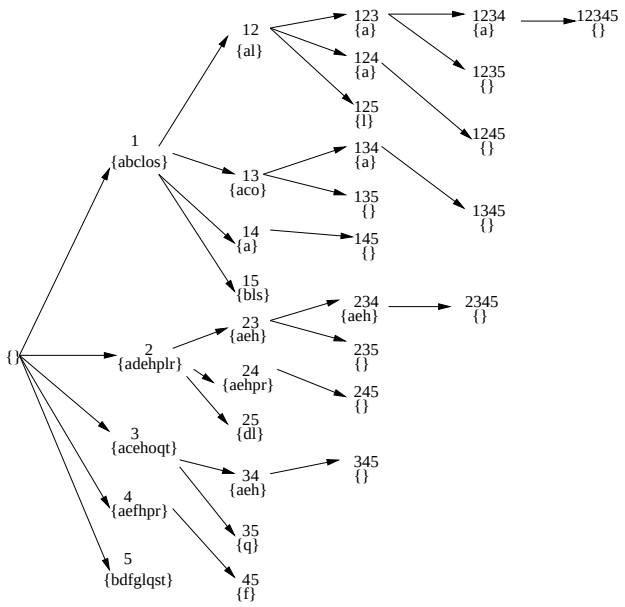

Figure 3: The Row Enumeration Tree.

complete row set enumeration tree without application of any pruning strategies. Each node in the tree represents a row set R'; also shown is $\mathcal{F}(R')$. For example, the node 12 represents the row set $\{r_1, r_2\}$, along with it supporting feature set $\mathcal{F}(12) = al$.

To find frequent closed patterns, CARPENTER performs a depth first search (DFS) of the row set enumeration tree. By imposing a total order, such as lexicographic order on the row sets, we are able to perform a systematic search for closed patterns. For example, DFS on the row enumeration tree in Figure 3 will be {1, 12, 123, 1234, 12345, 1235,...,45, 5} (in absence of any optimization and pruning strategies).

LEMMA 3.1. *Let F be a closed pattern and $\mathcal{R}(F)$ be the set of rows that contain F, then $\mathcal{R}(F)$ is unique. In other words, there does not exist a closed pattern F', $F' \neq F$, that satisfies $\mathcal{R}(F) = \mathcal{R}(F')$.*

Proof: *We prove by contradiction. Assume there exists a closed pattern F' that satisfies $\mathcal{R}(F) = \mathcal{R}(F')$ but $F' \neq F$. Let pattern $CF = F' \cup F$. Then $\mathcal{R}(CF) = \mathcal{R}(F) = \mathcal{R}(F')$. Since $F' \subset CF$, this contradicts the fact that F' is closed.* □

By Lemma 3.1, each closed pattern corresponds to a unique set of rows. By enumerating all combinations of rows as shown in Figure 3, we ensure that all closed patterns in the datasets are enumerated. It is obvious that a complete traversal of the row enumeration tree is not efficient and pruning techniques must be introduced to prune off unnecessary searches.

Let X be a subset of rows. Given the transposed table TT, a **X-conditional transposed table**, denoted as $TT|_X$, is a subset of tuples from TT such that: 1) For each tuple x in TT, there exist a corresponding tuple x' in $TT|_X$. 2) x' contains all rows in x with row ids larger than any row in X. As an example, let the transposed table in Figure 1(b) be TT and let $X = 23$. The X-conditional transposed table, $TT|_X$ is as shown in Figure 2.

Our formal algorithm is shown in Figure 4. For clarity, we assume that the database is already transposed with infrequent features removed. This pre-processing step is trivial

Algorithm CARPENTER
Input: Transposed table TT, features set F and support level $minsup$
Output: Complete set of frequent closed patterns, FCP
Method:

1. *Initialization.* $FCP = \emptyset$. Let R be the (numerically sorted) set of rows in the original table;
2. *Mine Frequent Closed Pattern.* MinePattern($TT|_\emptyset$,R, FCP);

Subroutine: MinePattern($TT'|_X$,R', FCP).
Parameters:

- $TT'|_X$: A X-conditional transposed table;
- R': A subset of rows which have not been considered in the enumeration;
- FCP: The set of frequent closed patterns that have been found;

Method:

1. Scan $TT'|_X$ and count the frequency of occurrences for each row, $r_i \in R'$. $Y = \emptyset$.
2. **Pruning 1:** Let $U \subset R'$ be the set of rows in R' which occur in at least one tuple of $TT'|_X$. If $|U|+|X| \leq minsup$, then return; else $R' = U$;
3. **Pruning 2:** Let Y be the set of rows which are found in every tuple of the X-conditional transposed table. *Let $R' = R' - Y$ and remove all rows of Y from $TT'|_X$*;
4. **Pruning 3:** If $\mathcal{F}(X) \in FCP$, then return;
5. If $|X| + |Y| \geq minsup$, add $\mathcal{F}(X)$ into FCP;
6. For each $r_i \in R'$,
   $R' = R' - \{r_i\}$
   MinePattern($TT'|_X|_{r_i}, R', FCP$);

Figure 4: The CARPENTER Algorithm

and take negligible time since our datasets usually fit in main memory. CARPENTER does recursive generation of conditional transposed tables, performing a depth-first traversal of the row enumeration tree. Each computed conditional table represents a node in the enumeration tree of Figure 3. For example, the 23-conditional table represents the node 23. After initializing FCP, the set of frequent closed pattern, to be empty and letting R to be the set of rows in the original table, CARPENTER calls the subroutine $MinePattern$ to recursively generate X-conditional tables.

The subroutine $MinePattern$ takes in three parameters $TT'|_X$, R' and FCP. TT'_X is a X-conditional table. R' contains the set of rows that will be used to enumerate the next level of conditional transposed table while FCP contains the frequent closed patterns that have been discovered so far.

Steps 1 to 4 in the subroutine perform the counting and pruning. They are extremely important for efficiency of CARPENTER . Before we explain these 4 steps, we will first show that the $MinePattern$ subroutine will only output a pattern if and only if it is a frequent closed patterns (in the absence of these 4 steps). This is done at Step 5 which checks whether $\mathcal{F}(X)$ is a frequent closed pattern before inserting $\mathcal{F}(X)$ into FCP, and at Step 6 which continues the next level of enumeration in the search tree. We prove the correctness of the two steps by two lemmas as follows:

LEMMA 3.2. *Let X be a subset of rows from the original table, then $\mathcal{F}(X)$ must be a closed pattern (not necessarily frequent).*
Proof: *We will prove by contradiction. Assuming $\mathcal{F}(X)$ is not a closed pattern, then there exists a feature f_i such that $\mathcal{R}(\mathcal{F}(X)) = \mathcal{R}(\mathcal{F}(X) \cup f_i)$. Since X contains all features of $\mathcal{F}(X)$, then $X \subset \mathcal{R}(\mathcal{F}(X))$. This means that f_i belongs to in every row in X, which contradicts the definition that $\mathcal{F}(X)$ is the maximal set of features common to rows of X.* □

Lemma 3.2 ensures that Step 5 only inserts closed patterns that are frequent into FCP. The main observation used in the proof is that $\mathcal{F}(X)$ cannot be a maximal feature set that is common in all rows of X unless it is a closed pattern. A check on $|X| + |Y|$ is needed to ensure that the minimum support threshold is satisfied (Note that Y is an empty set if the Steps 1 to 4 of $MinePattern$ are not executed). Together with Lemma 3.1, we know that the complete and correct set of frequent closed patterns will be in FCP.

LEMMA 3.3. $TT'|_X|_{r_i} = TT'|_{X \cup r_i}$ □

Lemma 3.3 is useful for explaining Step 6. It simply states that a $X \cup r_i$ conditional transposed table can be computed from a X conditional transposed table, $TT'|_X$, by selecting those tuples that contain r_i in $TT'|_X$. This is utilized in Step 6 where a recursive call on $MinePattern$ is called with $TT'|_X|_{r_i}$ as the conditional transposed table. This is in fact generating the $X + r_i$ conditional transposed table that is needed to represent the next level of row set enumeration.

Note that Step 6 implicitly represents a form of pruning too since it is possible to have $R' = \emptyset$. It can be observed from the enumeration tree that there exist some combinations of rows, X, such that $\mathcal{F}(X) = \emptyset$ (an example is node "134"). This implies that there is no feature which exists in all the rows in X. When this occurs, R' will be empty and no further enumeration will be performed.

We next look at the pruning techniques that are used in CARPENTER to enhance its efficiency. Our emphasis here is to show that our pruning steps do not prune off any frequent closed patterns, while preventing unnecessary traversal of the enumeration tree. This guarantees correctness.

The first pruning step is executed in Step 2 of $MinePattern$. The pruning is essentially aimed at removing search branches which can never yield closed patterns that satisfy the $minsup$ threshold. The following lemma is applied in the pruning.

LEMMA 3.4. *Let $TT'|_X$ be a X conditional transposed table. Let U be a set of rows which occur in at least one tuple of $TT'|_X$. If $|U| + |X| < minsup$, then it is* **not** *possible that for any $U' \subset U$, $\mathcal{F}(X \cup U')$ is a frequent closed pattern.*
Proof: *By definition, any row not in $TT'|_X$ cannot be combined with X to produce a non-empty closed pattern. Thus X can only be combined with some $U' \subset U$ in order to continue the enumeration. It is clear that the maximum support is bounded by $|U|+|X|$. If $|U|+|X| < minsup$, we can safely conclude that all the patterns in further enumeration will not be frequent.* □

In Step 3 of $MinePattern$, our second pruning strategy is applied. This pruning deals with rows that occur in all tuples of the X conditional transposed table. Such rows are immediately removed from $TT'|_X$ because of the following lemma

LEMMA 3.5. *Let $TT'|_X$ be a X conditional transposed table and Y be a set of rows which occur in every tuple of $TT'|_X$. Given any subset $R' \subset R$, we have $\mathcal{F}(X \cup R') = \mathcal{F}(X \cup Y \cup R')$.*
Proof: *By definition, $\mathcal{F}(X \cup R')$ contains a set of features which occur in every row of $X \cup R'$. Since the rows in Y occur*

in every tuple of $TT'|_X$, this means that these rows also occur in every tuples of $TT'|_{(X \cup R')}$ (Note: $TT'|_{(X \cup R')} \subset TT'|_X$). Thus, the set of tuples in $TT'|_{(X \cup R')}$ is exactly the set of tuples in $TT'|_{(X \cup R' \cup Y)}$. From this, we can conclude that $\mathcal{F}(X \cup R') = \mathcal{F}(X \cup Y \cup R')$. □

As an example to illustrate Lemma 3.5, let us consider the 23 conditional transposed table in Figure 2. Since row 4 occurs in every tuple of $TT|_{23}$, we can conclude that $\mathcal{F}(23)$=$\mathcal{F}(234)$ =aeh. Thus, we need not create $TT|_{234}$ in our search and row 4 need not be considered for further enumeration down that branch of the enumeration tree.

Our final and most complex pruning strategy is shown in Step 4 of $MinePattern$. This step will prune off any further search down the branch of node X if it is found that $\mathcal{F}(X)$ was already discovered previously in the enumeration tree. The inituitive reasoning which we will prove later is as follows: the set of closed patterns that will be enumerated from the descendants of node X must have been enumerated previously.

Unlike feature-based mining algorithms, such as CHARM and CLOSET, we need not perform detection of superset-subset relationship among the patterns since Lemma 3.2 already shows that only closed patterns will be enumerated in our search tree. For example, in Figure 3, it is not possible for the pattern $\{a, c\}$ to be enumerated although both $\{a\}$ and $\{a, c, o\}$ are closed patterns with support of 80% and 40% respectively. This is unlike CHARM and CLOSET, both of which will enumerate $\{a, c\}$ and check that it has the same support as a superset $\{a, c, o\}$ before discarding it as a non-closed pattern.

Another important thing to note here is that the correctness of the third pruning strategy (Step 3) **is dependent on the second pruning criteria**. This is essential because of the following lemma.

LEMMA 3.6. *Let X be the set of rows in the current search node and X' be the set of rows that result in $\mathcal{F}(X)$ (which is the same as F(X')) being inserted into FCP in earlier enumeration. If pruning strategy 2 is applied consistently in the algorithm, then the node representing X in the enumeration tree will not be the descendent of the node representing X' in the enumeration tree.*

Proof: *Assume otherwise, then $X' \subset X$. Let $Z = X - X'$. Since $\mathcal{F}(X) = \mathcal{F}(X')$, all rows in Z must be contained in all tuples of the X' conditional transposed table. Based on pruning strategy 2, the rows in Z would be added to X' and will be removed from subsequent transposed table down that search branch. Thus the node representing X will not be visited, which contradicts the fact that node X is currently being processed in the enumeration tree.* □

Consider the node 23 in Figure 3. As mentioned earlier, its descendant node 234 will not be visited since row 4 occurs in every tuple of 23-conditional transposed table. Without pruning strategy 2, this will not be the case.

We next try to prove that all branches from a node X in the enumeration tree can be pruned off if $\mathcal{F}(X)$ is already in FCP. We have the following lemma.

LEMMA 3.7. *Let $TT'|_X$ be the conditional transposed table in the current search node. Let X' be the set of rows which result in $\mathcal{F}(X)$ (which equals to $\mathcal{F}(X')$) being inserted into FCP in earlier enumeration. Let x_{f_i} and x'_{f_i} be the two tuples that represent feature f_i in $TT'|_X$ and $TT'|_{X'}$ respectively. We will have $x_{f_i} \subset x'_{f_i}$ for all $f_i \in \mathcal{F}(X)$.*

Proof: *We know that $\mathcal{F}(X) = \mathcal{F}(X')$ which implies that the set of features represented by tuples in both the conditional transposed tables are the same.*

Let the maximal set of rows that contains the feature set $\mathcal{F}(X')$ be $R'_{max} = \{r_1, ..., r_n\}$ which is sorted in numerical order. Let $X' = \{r'_1, ..., r'_m\}$ be the first row set that causes $\mathcal{F}(X')$ to be inserted into FCP and $X' = \{r'_1, ..., r'_m\}$ is also sorted in numerical order.

Based on Lemma 3.6, X cannot be a descendent of X' in the enumeration tree. Thus, X must be of the form $(X' - A) \cup B$ where $A \subset X'$ and $B \subset R'_{max} - X'$, $A \neq \emptyset$, $B \neq \emptyset$. By lexicographic row set search, we can conclude from here that there exists a row r'_i such that $i > m$ and $r'_i \in X$.

By definition of a conditional transposed table, we know that all rows which occur before r'_m will be removed in $TT'|_{X'}$. Likewise, all rows occurring before r'_i will be removed in TT'_X. Since $i > m$, a tuple x'_{f_i} representing feature f_i in $TT'|_{X'}$ will have less rows being removed than the corresponding tuple x_{f_i} representing feature f_i in $TT'|_X$. Hence the proof. □

In a less formal term, Lemma 3.7 shows that if X' is the first combination of rows that cause $\mathcal{F}(X')$ to be inserted into FCP, then the conditional transposed table $TT'|_{X'}$ will be more "general" than any other conditional transposed table $TT'|_X$ in which $\mathcal{F}(X) = \mathcal{F}(X')$. "General" in this case, refers to the fact that each tuple in $TT'|_{X'}$ is in fact a superset of the corresponding tuple in $TT'|_X$. We will now formalize our third pruning strategy as a theorem.

THEOREM 3.1. *Given a node representing a set of rows X in the enumeration tree, if $\mathcal{F}(X)$ is already in FCP, then all enumeration down that node can be pruned off.*

Proof *Let X' be the combination of rows that first cause $\mathcal{F}(X)$ to be inserted into FCP. From Lemma 3.7, we know that any tuple x'_{f_i} in the X' conditional table will be a superset of a corresponding tuple x_{f_i} in the X conditional table and X' conditional table has the same number of tuples with X conditional table.*

Since we know that the next level of search at node X in the enumeration tree is based on the set of rows in the X conditional transposed table, it is easy to conclude that the possible enumeration at the node X is a subset of the possible enumeration at node X'. Since X' had been visited, it is thus not necessary to perform any enumeration from the node X onwards. □

Consider the node 23 in Figure 3 which is the first node that results in the insertion of aeh based on the enumeration order. Thus, $R'_{min} = 23$. Next look at node 34. We have $\mathcal{F}(34) = \mathcal{F}(23) = aeh$. It can be seen that node 34 is not a descendant of node 23 and that 34 satisfies the formula $(R'_{min} - A) \cup B$, $A = 2$, $B = 4$. In this case, pruning strategy 3 can be applied and no further enumeration will be done from node 34.

CARPENTER is implemented by adopting the in-memory, pointer-based approach in BUC [3] [2]. With the in-memory pointer, CARPENTER does not construct conditional transposed table physically, thus saving space. Due to space limitation, we will not give details on the implementation of CARPENTER .

[2]We note also that there are other alternatives for implementation including building a FP-tree [4] on the transposed table and adopting the vertical data representation in the row-wise manner [10]. However, our central theme of row enumeration is independant of these techniques and we leave it to interested readers to explore them during their own implementation.

4. PERFORMANCE STUDIES

In this section, we compare the performance of CARPENTER against other algorithms. All experiments were performed on a PC with a Pentium III 1.4 Ghz CPU, 1GB RAM and a 80GB hard-disk. Algorithms were coded in Standard C.

Algorithms: We compare CARPENTER against two other closed pattern discovery algorithms, CHARM [10] and CLOSET [7] Experiments in [7, 10] have shown that depth-first mining algorithms like CHARM and CLOSET are substantially better than levelwise mining algorithms like Close[6] and Pascal [2]. To make a fair comparison, CHARM and CLOSET are also run in the main memory after one disk scan is done to load the datasets. The run time for CARPENTER includes the time for transposing the datasets.

Datasets: We choose 3 real-life gene/protein expression datasets to analyze the performance of CARPENTER . The **Lung Cancer (LC)** dataset [3] is a gene expression dataset. The rows in the dataset represent sample tissues and these tissues can come from either malignant pleural mesothelioma (MPM) or adenocarcinoma (ADCA) of the lung. There are 181 tissue samples and each sample is described by the activity level of 12533 genes or features. The **Acute Lymphoblastic Leukemia (ALL)** dataset [4] is also a gene expression dataset containing tissues from cancerous/non-cancerous cells. There are 215 tissue samples described by activity level of 12533 genes. The **Ovarian Cancer (OC)** dataset [5] is for identifying proteomic patterns in serum that distinguish ovarian cancer from non-cancer cases. There are 253 samples each described by activity level of 15154 proteins. These expression datasets have real valued entries, which have to be discretized to obtain binary features; we do an equal-depth partition for each attribute using 20 buckets. [6]

Parameters: Two parameters are varied in our experiment, *minimum support* (*minsup*) and *length ratio* (l). We use a default value of $minsup = 4\%$. The parameter *length ratio*, l, has a value between 0 and 1. It is used to generate new datasets with different average row size from the original datasets. A dataset with a length ratio of l retains on average $l * 100\%$ of the columns in the original dataset. Columns to be retained are randomly selected for each row. The default value used is $l = 0.6$, unless otherwise stated.

4.1 Varying Minimum Support

Figure 5 shows how CARPENTER compares against CHARM and CLOSET as *minsup* is varied with $l = 0.6$. Note that the y-axis is in logarithmic scale. There is a large variation in the running time for both CHARM and CLOSET even though the variation in absolute *minsup* is small. This is because the average length of each row after removing the infrequent features can increase (decrease) substantially due to a small decrease (increase) in *minsup* value. This increases (decreases) the search space of both CHARM and CLOSET substantially, resulting in a large difference in running time.

Among the three algorithms, we find that CLOSET is the slowest and has the steepest increases in run time as *minsup* is decreased. CHARM on the other hand is generally 2 to 3 orders of magnitude slower than CARPENTER and only outperforms CARPENTER at higher support level, where the difference in time is under 10 seconds.

4.2 Varying Length Ratio

Figure 6 shows the performance comparison of the methods as we vary l, with $minsup = 4$. The growth in run time of all the algorithms is exponential with respect to the length ratio (note the log scaled y-axes). CARPENTER is however substantially faster than CHARM and CLOSET. While CHARM outperforms CLOSET, CARPENTER can be up to a 100 times faster than CHARM and 1000 times faster than CLOSET.

As we can see, in all the experiments we conducted, CARPENTER outperforms CHARM and CLOSET in most cases. These results clearly demonstrate that CARPENTER is very efficient in finding frequent closed patterns on datasets with small number of rows and large number of features.

5. RELATED WORK

Frequent pattern mining [1, 5, 11, 8] as a vital topic has received a significant amount of attention during the past decade. The number of frequent patterns in a large data set can be very large and many of these frequent patterns may be redundant. To reduce the frequent patterns to a compact size, mining frequent closed patterns has been proposed. The followings are some new advances for mining closed frequent patterns.

Close [6] and Pascal [2] are two algorithms which discover closed patterns by performing breadth-first, column enumeration. Close [6] is an Apriori-like algorithm to determine a closed itemset. Due to the level-wise approach of Close and Pascal, the number of feature sets enumerated will be extremely large when they are run on long biological datasets.

In [7], the CLOSET algorithm was proposed for mining closed frequent patterns. Unlike Close and Pascal, CLOSET performs depth first, column enumeration. CLOSET uses a novel frequent pattern tree (FP-structure) for a compressed representation of the datasets. It then performs recursive computation of conditional tables to simulate the search on the column enumeration tree. CLOSET is unable to handle long biological datasets because of two reasons. First, the FP-tree is unable to give good compression for long rows. Second, there are too many combinations when performing column enumerations. CLOSET+ [9] is a recent improvement on CLOSET. Our study show that CARPENTER still outperforms CLOSET+ on average by around 500-600 times. [7]

Another algorithm for mining frequent closed pattern is CHARM [10]. Like CLOSET, CHARM performs depth-first, column enumeration. However, unlike CLOSET, CHARM stores the dataset in a vertical format where a list of row ids is stored for each feature. These row id lists are then merged during the column enumeration to generate new rows id lists that represent nodes in the enumeration tree. In addition, a technique called *diffset* is used to reduce the size of the row id lists and the computational complexity for merging them. Although performance studies in [10] shows that CHARM is substantially faster than all other algorithm on most datasets, CHARM is still unable to handle long biolog-

[3] available from http://www.chestsurg.org
[4] http://www.stjuderesearch.org/data/ALL1/
[5] http://clinicalproteomics.steem.com/
[6] Fewer buckets results in a extremely high running time (up to a few days) for CHARM and CLOSET.

[7] We will like to thank Jianyong Wang and Jiawei Han for making the executable code of CLOSET+ available to us. They have indicated that the version of CLOSET+ that they pass to us at press time is not optimized for our datasets and an optimized version will be released in the future.

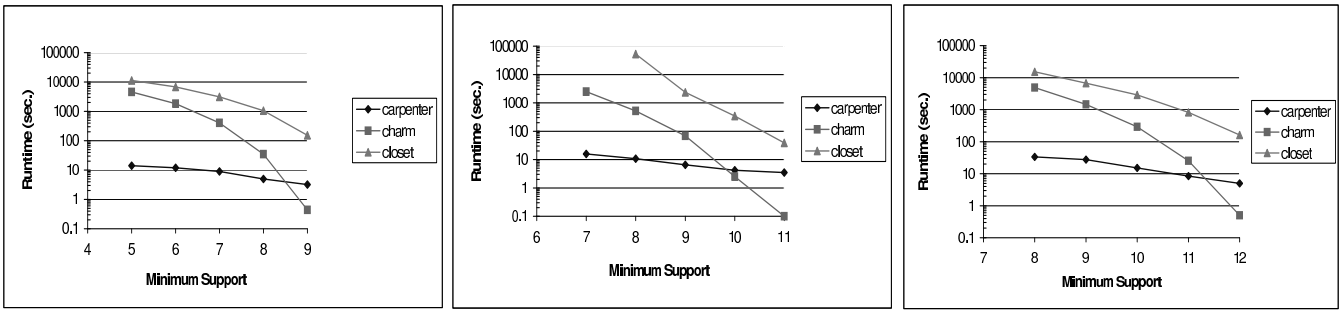

Figure 5: Varying *minsup* with l=0.6: a) LC, b) ALL, c) OC

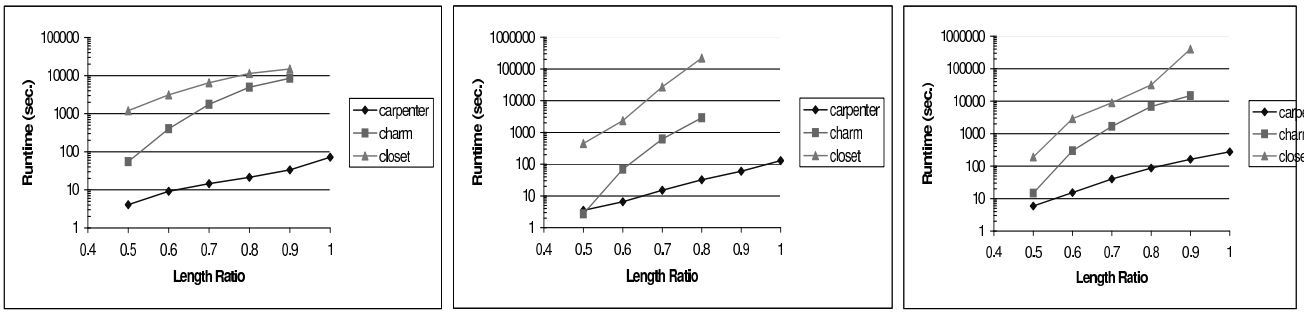

Figure 6: Varying l with $minsup = 4\%$: a) LC, b) ALL, c) OC

ical dataset efficiently because it performs feature enumeration.

6. CONCLUSION

In this paper, we proposed an algorithm called CARPENTER for finding frequent closed patterns in long biological datasets. CARPENTER makes use of the special charateristic of biological datasets to enhance its efficiency. It adopts the novel approach of performing row enumeration instead of the conventional column enumeration so as to overcome the extremely high dimensionality of many biological datasets. Experiments show that this bold approach yields good payoff as CARPENTER outperforms exisiting closed pattern discovery algorithms like CHARM and CLOSET by a large order of magnitude when they are running on long biological datasets. In the future, we will look at how CARPENTER can be extended to work on other datasets by using a combination of column and row enumerations.

7. REFERENCES

[1] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proc. 1994 Int. Conf. Very Large Data Bases (VLDB'94)*, pages 487–499, Santiago, Chile, Sept. 1994.

[2] Y. Bastide, R. Taouil, N. Pasquier, G. Stumme, and L. Lakhal. Mining frequent closed itemsets with counting inference. In *SIGKDD Explorations, 2(2)*, Dec. 2000.

[3] K. Beyer and R. Ramakrishnan. Bottom-up computation of sparse and iceberg cubes. In *Proc. 1999 ACM-SIGMOD Int. Conf. Management of Data (SIGMOD'99)*, pages 359–370, Philadelphia, PA, June 1999.

[4] J. Han, J. Pei, and Y. Yin. Mining partial periodicity using frequent pattern trees. In *Computing Science Techniqcal Report TR-99-10*, Simon Fraser University, July 1999.

[5] H. Mannila, H. Toivonen, and A. I. Verkamo. Efficient algorithms for discovering association rules. In *Proc. AAAI'94 Workshop Knowledge Discovery in Databases (KDD'94)*, pages 181–192, Seattle, WA, July 1994.

[6] N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal. Discovering frequent closed itemsets for association rules. In *Proc. 7th Int. Conf. Database Theory (ICDT'99)*, pages 398–416, Jerusalem, Israel, Jan. 1999.

[7] J. Pei, J. Han, and R. Mao. CLOSET: An efficient algorithm for mining frequent closed itemsets. In *Proc. 2000 ACM-SIGMOD Int. Workshop Data Mining and Knowledge Discovery (DMKD'00)*, pages 11–20, Dallas, TX, May 2000.

[8] P. Shenoy, J. Haritsa, S. Sudarshan, G. Bhalotia, M. Bawa, and D. Shah. Turbo-charging vertical mining of large databases. In *Proc. 2000 ACM-SIGMOD Int. Conf. Management of Data (SIGMOD'00)*, pages 22–23, Dallas, TX, May 2000.

[9] J. Wang, J. Han, and J. Pei. Closet+: Searching for the best strategies for mining frequent closed itemsets. In *Proc. 2003 ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD'03)*, Washington, D.C., Aug 2003.

[10] M. Zaki and C. Hsiao. Charm: An efficient algorithm for closed association rule mining. In *Proc. of SDM 2002*, 2002.

[11] M. J. Zaki, S. Parthasarathy, M. Ogihara, and W. Li. New algorithms for fast discovery of association rules. In *Proc. 1997 Int. Conf. Knowledge Discovery and Data Mining (KDD'97)*, pages 283–286, Newport Beach, CA, Aug. 1997.