Mining Maximal Hyperclique Pattern: A Hybrid Search Strategy

Yaochun Huang [a,*], Hui Xiong [b], Weili Wu [a], Ping Deng [a], Zhongnan Zhang [a]

[a] *University of Texas - Dallas, Department of Computer Science*
Email: {yxh038100, weiliwu, pxd010100, zxz022000}@utdallas.edu,

[b] *Rutgers University, MSIS Department*
Email: hui@rbs.rutgers.edu

Abstract

A hyperclique pattern is a new type of association pattern that contains items which are *highly affiliated* with each other. Specifically, the presence of an item in one transaction strongly implies the presence of every other item that belongs to the same hyperclique pattern. In this paper, we present an algorithm for mining maximal hyperclique patterns, which specifies a more compact representation of hyperclique patterns and are desirable for many applications, such as pattern-based clustering. Our algorithm exploits key advantages of both the Depth First Search (DFS) strategy and the Breadth First Search (BFS) strategy. Indeed, we adapt the equivalence pruning method, one of the most efficient pruning methods of the DFS strategy, into the process of the BFS strategy. Our experimental results show that the performance of our algorithm can be orders of magnitude faster than standard maximal frequent pattern mining algorithms, particularly at low levels of support.

Key words: Association Rules, Hyperclique Patterns

* Corresponding author.

1 Introduction

The association-rule mining problem [3, 2] is concerned with finding relationships among items in a large-scale data set. In the past decade, the association-rule mining has been the subject of extensive research in data mining [1, 3, 2, 4, 10, 14, 11]. Given a set of transactions, the objective of the association-rule mining is to extract all rules of the form $X \Rightarrow Y$, where X and Y are sets of items, which satisfy user-specified minimum support and minimum confidence thresholds. Support measures the fraction of transactions that obey the rule, while confidence provides an estimate of the conditional probability that a transaction contains Y, given that it contains X. Both metrics are useful because they provide an indication of the strength and statistical significance of an association rule.

Standard association-rule mining algorithms have the emphasis on discovering frequent patterns. However, these approaches may lose efficiency when the support threshold is low. Also, frequent patterns usually contain objects which are weakly related to each other [18]. Instead, a hyperclique pattern [18] was proposed as a new type of association patterns that contain items that are *highly affiliated* with each other. Specifically, the presence of an item in one transaction strongly implies the presence of every other item that belongs to the same hyperclique pattern. The h-confidence measure captures the strength of this association and is defined as the minimum confidence of all association rules of an itemset. An itemset is a **hyperclique pattern** if the h-confidence of this pattern is greater than a user-specified minimum h-confidence threshold. A hyperclique pattern is a **maximal hyperclique pattern** if no superset of this pattern is a hyperclique pattern.

Maximal hyperclique patterns specify a more compact representation of hyperclique patterns and are desirable in many application domains, such as pattern preserving clustering [17], which can easily produce interpretable clustering results. However, to our best knowledge, there are no efficient algorithms for mining maximal hyperclique patterns in the literature. As a result, the objective of this paper is to design an efficient algorithm for mining maximal hyperclique patterns in large-scale data sets.

In general, for the association pattern mining, there are two search strategies: Breadth First Search (BFS) and Depth First Search (DFS). The BFS strategy performs pattern search in a level-wise manner. In other words, it first discovers all the size-1 patterns at level 1, followed by all the size-2 patterns at level 2, and so on, until no pattern is generated at a particular level. If mining maximal hyperclique patterns using the BFS strategy, we could apply *Prevalence Pruning*; that is, an itemset can be pruned if one of its subset is not a hyperclique pattern. This pruning is based on the anti-monotone property of support and h-confidence measures. The limitation of this strategy is that we need to generate all the subsets of a maximal hyperclique pattern. In contrast, the DFS strategy avoids generating all the intermediate patterns and can directly find maximal hyperclique patterns. For the DFS strategy, a lot of pruning methods, such as equivalence pruning, leftmost pruning, full pruning, and dynamic ordering [4, 6, 20], can be applied. However, the DFS strategy cannot apply the *Prevalence Pruning* method, since we do not generate all subsets of a candidate pattern for this strategy.

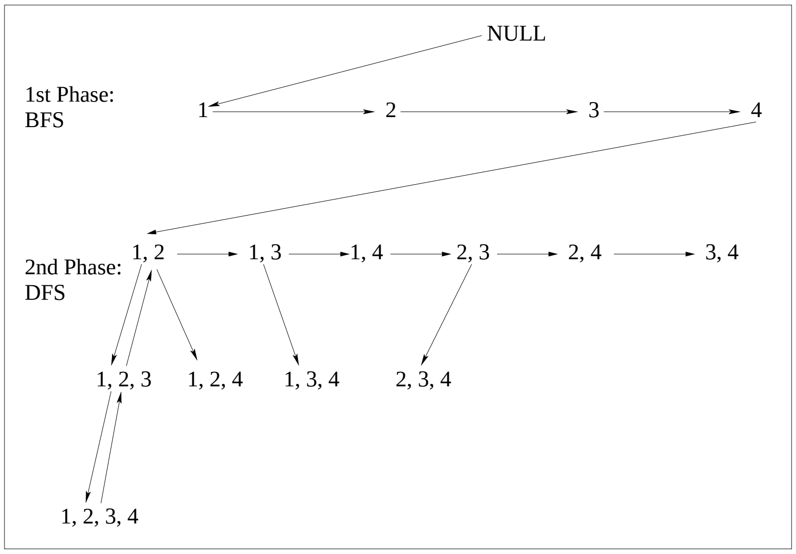


Fig. 1. An Illustration of the Hybrid Mining Method.

In this paper, we exploit key advantages of both the DFS strategy and the BFS strategy and design a hybrid **M**aximal **H**yperclique **P**attern (MHP) mining algorithm. Figure 1 illustrates our MHP algorithm, which has two phases. In the first BFS phase, for a given depth L, we use the Apriori-like approach [2] to generate all the size-L hyperclique patterns. In the second phase, the MHP algorithm takes the DFS search strategy. All the DFS pruning methods are used in this phase. Also, since we have all the size-L hyperclique patterns, an itemset can be pruned by prevalence pruning method if any of its size-L subset has not been generated. Considering the DFS strategy is much more efficient than the BFS strategy for finding maximal patterns and the major computation savings of the DFS strategy is due to the equivalent pruning method [1, 6, 20], we adapt the equivalent pruning method into our algorithm. In addition, we prove the correctness and completeness of our MHP algorithm. Finally, our experimental results show that the MHP algorithm can be orders of magnitude faster than maximal frequent pattern mining algorithms, such as MAFIA [6], particularly at low level of support.

Related works:

Agrawal et al. [3, 2] have proposed the classical DFS and BFS algorithm to discover frequent patterns. Some other researchers proposed the concepts of maximal [6, 4] and closed frequent patterns [20]. These frequent pattern mining algorithms are not very effective for identifying patterns at a low level of support.

Recently, there has been growing interest in developing techniques for mining association patterns without support constraints. For example, Wang et al. proposed the use of universal existential upward closure property of confidence to extract association rules without specifying the support threshold [15]. Cohen et al. have proposed using the Jaccard similarity measure, $sime(x, y) = \frac{P(x \cap y)}{P(x \cup y)}$, to capture interesting patterns without using a minimum support threshold [7]. Also, many researchers developed alternative techniques to push various types of constraints into the mining algorithm [5, 9, 12]. These approaches greatly reduce the number of patterns generated and improve computational performance by introducing additional constraints, but fail to offer any specific mechanism to eliminate weakly-related patterns involving items with different support levels.

Xiong et al. [18] introduced the concept of hyperclique patterns, which include items strongly related with each other. An h-confidence measure was used to identify hyperclique patterns. This measure possesses the desired anti-monotone and cross-support properties, which can be helpful for identifying strongly correlated items even at low levels of support. In this paper, we focus on finding maximal hyperclique patterns.

When Agrawal et al. [3] proposed the problem of association-rule mining, they provided an Apriori algorithm for mining frequent patterns. The Apriori is a basic BFS approach. Later on, Agarwal et al. [1] designed a more advanced BFS algorithm, called tree projection. Several optimal methods have been implemented in this BFS algorithm. They also mentioned the possibility of a hybrid searching strategy, but have not put the idea into practice. Compared to the BFS strategy, the DFS strategy is more fit to find maximal frequent patterns. There are many optimal pruning methods which prune non-maximal patterns to reduce the searching space [4, 6, 20]. Burdick et al. [6] analyzed the performances of these methods and showed that the *equivelant pruning method* could greatly speed up the processing procedure. Han et al. [10] constructed a very compact structure, FP tree, to store the information of patterns. Avoiding generating candidate patterns, FP tree could extract maximal patterns more efficiently. Zaki [19] analyzed the performances of top-down and bottom-up searching strategies for mining maximal frequent patterns and designed Clique and Eclat algorithms for this purpose.

Overview: The remainder of this paper is organized as follows. Section 2 defines some basic concepts. In section 3, we propose a framework for mining maximal hyperclique patterns. We describe the algorithm details and prove the correctness and completeness of the algorithm in Section 4. Our experimental results are presented in Section 5. Finally, in section 6, we draw conclusions and suggest future work.

2 Basic Concepts

To facilitate our discussion, we first present some basic concepts in this section.

Definition 1 *The* **h-confidence** *of an itemset $P = \{i_1, i_2, \cdots, i_m\}$, denoted as $hconf(P)$, is a measure that reflects the overall affinity among items within the itemset. This measure is defined as $min\{conf\{i_1 \rightarrow i_2, \ldots, i_m\}, conf\{i_2 \rightarrow i_1, i_3, \ldots, i_m\}, \ldots, conf\{i_m \rightarrow i_1, \ldots, i_{m-1}\}\}$, where $conf$ is the traditional definition of association rule confidence [2].*

Definition 2 *An itemset P is a* **hyperclique pattern** *if $support(P) \geq \theta$ and $hconf(P) \geq H_c$, where θ is a user-specified minimal support threshold and H_c is a user-specified minimal h-confidence threshold. When the h-confidence threshold equals to 0, hyperclique patterns become frequent patterns.*

Definition 3 *For a hyperclique pattern, HP, if none of its supersets is a hyperclique pattern, we say HP is a* **Maximal Hyperclique Pattern (MHP)**. *This means, a pattern $P \in$ **MHP** $\Longleftrightarrow$ **P** $\in$ **HP** and $\forall$ **P'**$\supset$ **P**, **P'** $\notin$ **HP**.*

Definition 4 *The* **order** *of items: for two items i_1 and i_2, if $support(i_1) \leq support(i_2)$ and the name of i_1 is preceding of the name of i_2 in the lexicographic order, we say i_1 is* **lexicographic before** *i_2. This can also be denoted as $i_1 < i_2$.*

In the rest of this paper, we arrange items in each pattern in order, unless otherwise noted.

Definition 5 *The* **order** *of patterns: for two different patterns $P_1 = \{i_1, i_2, ...i_k\}$ and $P_2 = \{i'_1, i'_2, ..., i'_l\}$, if $(P_1 \subset P_2) \vee (\exists\ m, m < k$ and $m < l, \forall n, 1 \leq n \leq m-1, i_n = i'_n$ and $i_m < i'_m)$, we say P_1* **lexicographic before** *P_2. It can also be denoted as $P_1 < P_2$.*

3 A Framework for Mining Maximal Hyperclique Patterns

In this section, we present a framework of two-phase maximal hyperclique pattern mining. In the first BFS phase, we retrieve all the size-L hyperclique patterns. In other words, the first L levels of the *lexicographic tree* [13] will be searched using Apriori-like methods [2]. In the second phase, we apply the DFS strategy to extract all the Maximal Hyperclique Patterns (MHP).

For better illustration, we construct a small demo dataset. Table 1 shows this sample data set and Table 2 shows the support of items in the sample data set. For a minimum *Support Threshold* (θ)= 0.15, and a minimum $H-confidence$ *Threshold* (H_c)= 0.55, Figure 2 illustrates the two-phase maximal hyperclique pattern mining process on the sample data set.

3.1 Basic Definitions

For a pattern, there are three concepts related to items of this pattern: the item set, the equivalence item set, and the tail item set. We first introduce these three concepts.

Definition 6 *The* **Item Set** *(P.item) of a pattern P is the set of all the items in the pattern.*

Definition 7 *The* **Tail Item Set** *(P.tail) of a pattern is the set of items which can be used to generate the super pattern of this pattern in the DFS phase.*

In the DFS phase, we retrieve all the patterns by generating the super patterns of a given pattern(P) with its tail items [13, 1]. All tail items are included in P.tail. As can be seen in Figure 2, {3,4,7} is a hyperclique pattern, and items 8 and 9 could be used to generate super patterns of {3,4,7}, since all the size-3 sub patterns of {3,4,7,8} and {3,4,7,9} are hyperclique patterns. So the *tail item set* of {3,4,7} is {8,9}.

Definition 8 *For a pattern P, if an item appears in all the transactions that contain P.item, but not in P.item, we say that this item is an* **equivalence item** *with P.*

For instance, item 5 always appears in every transaction which includes pattern {1,2}. So, 5 is an **equivalence item** of pattern {1,2}.

Lemma 1 *If an item is an equivalence item of a pattern, it should also be an equivalence item of its super patterns.*

If an item i is an equivalence item of P, but $\frac{support(P)}{support(i)} < H_c$, the union of $\{i\}$ and P.item is not a hyperclique pattern. We can prune this kind of equivalence items.

Definition 9 *If an item is an* **equivalence item** *of a pattern P, and the*

Table 1
A Sample Data Set

TID	Items
1	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11
2	3, 4, 7, 8, 9, 11
3	3, 4, 5, 6, 7, 8, 9, 11
4	1, 2, 3, 4, 5, 6, 7, 8, 9, 11
5	1, 3, 4, 7, 8, 9
6	2, 3, 4, 7, 8
7	3, 4, 7, 9
8	3, 4, 8, 9
9	3, 7, 8, 9
10	4, 7, 8, 9

Table 2
Support of Items in the Sample Data Set

Item	TID	Support
1	1, 4, 5	0.3
2	1, 4, 6	0.3
3	1, 2, 3, 4, 5, 6, 7, 8, 9	0.9
4	1, 2, 3, 4, 5, 6, 7, 8, 10	0.9
5	1, 3, 4	0.3
6	1, 3, 4	0.3
7	1, 2, 3, 4, 5, 6, 7, 9, 10	0.9
8	1, 2, 3, 4, 5, 6, 8, 9, 10	0.9
9	1, 2, 3, 4, 5, 7, 8, 9, 10	0.9
10	1	0.1
11	1, 2, 3, 4	0.4

$Support\ Threshold(\theta)=0.15$

$H-confidence\ Threshold(H_c)=0.55$

union of this item and P.item is also a hyperclique pattern, we say this item is a Pure Equivalence Item, **PE item**, *of the pattern P.*

In the example dataset, we know both item 4 and 6 are equivalence items of {1,2}. Hconf({1,2,4})=0.22< H_c, and Hconf({1,2,5})=0.66> H_c, so 5 is a PE item of {1,2}, but 4 is not.

If we generate the closed frequent itemset or maximal frequent itemset with the DFS approach, the $EquivalencePruning$ method could move the PE item from P.tail to the P.item directly [6, 20]. However, this method may break the limitation of h-confidence when we generate super patterns. As shown in the sample data set, item 11 is a PE item of {5}, but not a PE item for {1, 5}. In other words, item 11 cannot be added into {5}, since we do not know whether item 11 is a PE item of the super patterns of {5} or not. Also, we apply the equivalence pruning in the BFS phase. Since adding items will change the size of patterns, we need to maintain a set of PE items for patterns.

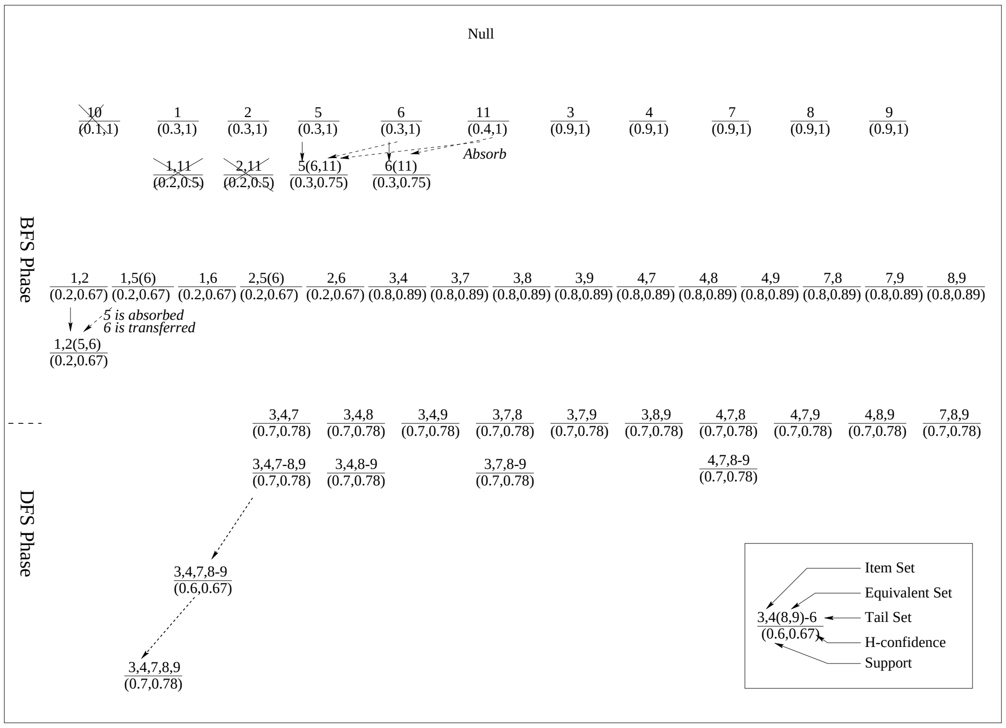


Fig. 2. An Illustration of the Two-Phase Maximal Hyperclique Pattern Mining

Definition 10 *The* **Equivalence Item Set** *of a pattern, P.equivalence, is the item set of all PE items of the pattern P.*

If we find an item is a PE item of a pattern P, we could add it to P.equivalence. While the super patterns of P are generated, they will succeed their own PE items from P.equivalence. In this case, the items of P are separated in P.item and P.equivalence, and the real pattern of P should be P.item $\cup$ P.equivalence. In Figure 2, items 5 and 6 are PE items of {1,2}, so the *item set* of {1,2(5,6)} is {1,2}, and the *equivalence item set* is {5,6}.

Definition 11 **Union** *of a pattern, P.union, is P.item∪P.equivalence. P.union is the real itemset of P. Indeed, support(P.union)=support(P.item).*

Definition 12 **Size** *of a Pattern P: we define the size of P as the size of P.item, no matter how many items in P.equivalence.*

For instance, the union of pattern {1,2(5,6)} are $\{1,2\} \bigcup \{5,6\} = \{1,2,5,6\}$. However, the size of {1,2(5,6)} is 2, which is the same as {1,2}.

Definition 13 **Sub pattern***: For two pattern P_1, P_2, if P_1.item is a subset of P_2.item, we say that P_1 is a* **sub pattern** *of P_2, even P_1.union is not a subset of P_2.union, and P_2 is a* **super pattern** *of P_1. If the size of P_1 is smaller* *than the size of P_2, P_1 is a* **pure sub pattern** *of P_2.*

In the sample data set, {1,5(6)} is a super pattern of both {5(6,11} and {1,5}. However, only {5(6,11)} is a pure sub pattern of {1,5(6)}, since size({5(6,11)}) < size({1,5(6)}), and size({1,5}) = size({1,5(6)}),

Definition 14 *For a Pattern P_1, if i' is P_1's equivalence item and all the items in HP_1.item are lexicographic before item i', we say item i' is a* **Pro equivalence item** *of P_1.*

Definition 15 *For a Hyperclique Pattern HP_1, if item i' is both HP_1's PE item and Pro equivalence item, the item i' is a Pro Pure equivalence item (***PPE item***) of HP_1.*

For instance, in the sample data set, the two equivalence items of pattern {1,2}, 4 and 5, are pro equivalence items of {1,2}, since items 1 and 2, are lexicographic before 4 and 5.

3.2 Pruning Methods in the BFS phase

At the initial stage, the algorithm generates the size-1 patterns and counts the support of these patterns. All items which have supports less than the user-specified support threshold are pruned. Meanwhile, these items are sorted during this stage. For instance, consider the example dataset shown in Table 1, item 10 can be pruned since $support(10) \leq \theta$. Also, as shown in Figure 2, the algorithm constructs the size-1 hyperclique patterns and sort all items in lexicographic order:{1}{2}{5}{6}{11}{3}{4}{7}{8}{9}.

In the BFS phase, the algorithm exploits an apriori-like approach to generate the **size-L** hyperclique patterns from **size-(L-1)** hyperclique patterns. There are three pruning strategies applied in this phase as follows.

Prevalence Pruning. For an Apriori-like algorithm, a size-k pattern P_k with $P_k.item = \{i_1, i_2, ..., i_k\}$ is generated by joining two size-(k-1) patterns:P_{k-1} and P'_{k-1}, $P_{k-1}.item = \{i_1, i_2, ..., i_{k-1}\}$ and $P'_{k-1}.item = \{i_1, i_2, ..., i_{k-2}, i_k\}$. If P_{k-1} and P'_{k-1} exist, the algorithm first checks whether all the other size-(k-1) sub patterns of P_k exist. If one of the sub patterns does not exist, P_k is not a hyperclique pattern and can be pruned [1].

H-confidence Pruning.

Before generating a size-k pattern P_k, we could calculate the ratio: $\frac{support(HP_{k-1})}{support(i_k)}$. If this ratio is less than h_c, hconf(P_k) should be also less than H_c, since support(P_k) $\leq$ support(P_{k-1}) [18]. For instance, as shown in Figure 2, support(1)=0.3, support(3)=0.9, hconf({1,3})=$\frac{support(\{1,3\})}{support(3)} \leq \frac{support(\{1\})}{support(3)} < 0.34 < H_c$, therefore the pattern {1,3} is pruned.

Equivalence Pruning. We apply the *Equivalence Pruning* method to reduce the number of patterns generated. If support(HP_k)=support(HP_{k-1}), i_k should be a PPE item of HP_{k-1}, and be **absorbed** into HP_{k-1}.equivalence. For example, in Figure 2, support({5, 6})=support({5})=0.3, we add item 6 to {5}.equivalence and prune {5, 6}.

When generating a size-k hyperclique pattern HP_k, if the items in the equivalence sets of size-(k-1) sub hyperclique patterns are PE items of HP_k, HP_k can **succeed** these items to its own equivalence set. For instance, in Figure 2, both {1, 5} and {2, 5} succeed item 6 from {5}.equivalence, but do not succeed item 11 since it would break the limitation of h-confidence.

When HP_{k-1} absorbing item i_k, all the equivalence items of the other size-(k-1) patterns-$\{i_2, i_3, ..., i_k\}, \{i_1, i_3, ..., i_k\}, \{i_1, ..., i_{k-1}, i_k\}$, are also equivalence items of HP_{k-1}. HP_{k-1} could **transfer** these items to HP_{k-1}.equivelance if they are PE items. In Figure 2, while generating the pattern {1, 2, 5} from {1, 2}, {1, 5} and {2, 5}, the pattern {1, 5} will absorb item 5, and transfer item 11 from {1, 5}.equivalence.

Indeed, when generating HP_k, if item i_k is in HP_{k-1}.equivalence, it is unnecessary to generate the HP_k, but transfer the PE items in the other size-(k-1) patterns' equivalence set to HP_{k-1}.equvialence.

After generating the size-k hyperclique patterns, we could check all the size-(k-1) hyperclique patterns in lexicographic order. For a size-(k-1) pattern HP_{k-1}, if its union is not a subset of any size-k pattern's union, it will be impossible to generate a hyperclique pattern whose union is the superset of HP_{k-1}.union in the following process. If this union is not a subset of an itemset in current **Maximal Hyperclique Pattern Set (MHPS)** either, this union is a maximal hyperclique pattern and could be added to the MHPS. For example, in Figure 2, after generating the size-2 patterns, it is found that the union of {1, 2} is {1, 2, 5, 6}, and no superset in either size-3 patterns' union or MHPS. Hence, the algorithm adds the union into MHPS. For pattern {1, 5}, the union of this pattern is {1, 5, 6}, and this pattern has no superset in size-3 patterns' union, but has a superset in MHPS, hence this pattern is pruned.

3.3 Pruning Methods in the DFS phase

In the BFS phase, the algorithm has identified all the size-L hyperclique patterns. At the beginning of the DFS phase, the algorithm adds the tail items to the tail sets of these patterns. For a size-L hyperclique pattern

HP, HP.item=$\{i_1, i_2, ..., i_L\}$, if there is an item i' such that: (1) item $i' \notin HP$.equivalence, (2) all the items in HP.item are lexicographic before i', and (3) all the size-L sub patterns of $\{i_1, i_2, ..., i_L, i'\}$ have been generated, the algorithm adds item i' to HP.tail. For instance, in Figure 2, item 8 and 9 are added to {3, 4, 7}'s tail set.

The super patterns of a hyperclique pattern(HP) are generated with the item in HP.tail, and succeed the PE item from HP.equivalence.

Equivalence Pruning.

Similar to the BFS phase, if a tail item i' is a PE item, we will add i' to the equivalence set of the pattern. If the size-1 pattern $\{i'\}$'s equivalence set is not null, the super patterns will succeed PE items from this set.

Full Pruning.

When we process the Pattern HP, if the union of HP.item, HP.equivalence and HP.tail is a subset of a pattern in current MHPS, all of the patterns generated by HP cannot be MHP since they have a super Hyperclique Pattern. We could prune this pattern directly. In Figure 2, when we process {3, 7, 8}, which tail set is {9}, {3, 4, 7, 8, 9} has already been added to MHPS. We will find $\{3, 7, 8\} \cup \{9\}$ is a subset of {3, 4, 7, 8, 9}, and prune {3, 7, 8}.

LeftMost Pruning.

When processing a hyperclique pattern HP, if the pattern at the end of this path is found to be MHP, all the patterns in the other paths should not be MHP. In this case, we could skip these patterns [6]. For example, in Figure 2, the end of left most path of {3, 4, 7} is {3, 4, 7, 8, 9}, and we find this pattern is MHP, we can skip all the other paths of {3, 4, 7}, and continue to process the next pattern.

Dynamic Reordering.

Bayardo et al. [5] showed that the benefit of dynamically reordering super patterns of a pattern is important. The performance can be 2 to 4 times faster. In our algorithm, we sort the super patterns in the increasing order of their support.

H-confidence Pruning. Similar to the BFS phase, for a tail item (i') of a hyperclique pattern(HP), if $\frac{support(HP)}{support(i')} < H_c$, we could prune i' from HP.tail.

Prevalence Pruning. Since we have generated the size-L hyperclique patterns in the BFS phase, for a hyperclique pattern HP, if one of its size-L sub-pattern is not generated, we can prune this pattern.

In the DFS phase, if a hyperclique pattern cannot generate any super hyperclique pattern, or none of these super hyperclique patterns could succeed all the items in its equivalence set, it will be impossible to find a super union of this pattern's union in the future. We will check this union with MHPS. If there is no super pattern in MHPS, we will add the union to MHPS.

```
MHS ALGORITHM
Input:      (a) $P = \{$A Pattern$\}$
            (b) $Data = \{$A DataSet represent a set of transaction $\}$
            (c) $\theta$: A minimal support threshold
            (d) $H_c$: A minimal h-confidence threshold
            (e) $L$: The retrieve level of the BFS phase
Output:     (1) A set of Maximal Hyperclique Patterns(MHPS)
            with support $\geq \theta$, hconf $\geq H_c$, and its superset without both
            such two properties.
Variables: k: the itemset size
            $HP_k$: a set of size-$k$ hyperclique patterns.
            $CMHP_k$: a set of size-$k$ candidate maximal hyperclique patterns.
            $MHPS$: set of maximal hyperclique patterns.
            $P_{super}$: a set of superset generated from $P$
Phase I: generate Hyperclique Patterns by BFS
 1.         $HP_1$= Initial($CP_1$, $\theta$, $H_c$, $Data$);
 2.         for (k=1;$k < L$;k++) do
 3.             $CP_{k+1}$ = Generate_and_Prune_Super($HP_k$, $\theta$, $H_c$);
 4.             $CMHP_k$ = set of patterns in $HP_k$ without superset union in $HP_{k+1}$;
 5.             Check_and_Add($CMHP_k$,$MHPS$);
Phase II: extract Maximal Hyperclique Patterns from $HP_L$ by DFS
 6.         Append_Tail($HP_L$);
 7.         for $\forall$ $P$ $in$ $HP_L$
 8.             Extract_MHP($P$);
Function Extract_MHP(Pattern $P$)
 9.         $P_{super}$=Generate_and_Prune_Super($P$, $\theta$, $H_c$, $HP_L$);
 10.        Sort_and_Append_Tail($P_{super}$)
 11.        for $\forall$ item $P_i \in P_{super}$
 12.            Extract_MHP($P_i$);
 13.        if $P$.union hasn't a super union in $P_{super}$
 14.            Check_and_Add($P$,MHPS);
```

Fig. 3. The Overview of the MHP Algorithm

4 The Maximal Hyperclique Pattern Mining Algorithm

4.1 Algorithm Description

Figure 3 shows an overview of the hybrid Maximal Hyperclique Pattern (MHP) mining algorithm, which has two phases: the Breadth First Search (BFS) phase and the Depth First Search (DFS) phase. In the BFS phase, Initial Function generates the size-1 hyperclique patterns, and items are sorted by support in non-decreasing order. In Generate_and_Prune_Super Function, the prevalence pruning, h-confidence pruning, and equivalence pruning are applied to prune the search space and size-k hyperclique patterns are generated from size-(k-1) hyperclique patterns. After extracting the size-k patterns, the algorithm extracts all size-(k-1) hyperclique patterns which have no super union in size-k hyperclique patterns to $CMHP_{k-1}$. In Check_and_Add Function, the algorithm checks the patterns in $CMHP_{k-1}$. If their unions are not subsets in MHPS, these unions are added into MHPS.

In the DFS phase, the Append_Tail Function generates the tail itemsets of size-L patterns. Extract_MHP is the major function for DFS mining. The traditional optimal methods, such as full pruning, leftmost pruning, and equivalence pruning, as well as new methods including prevalence pruning and h-confidence pruning, are implemented in Function Generate_and_Prune_Super. The Sort_and_Append_Tail Function implements the dynamic sorting and adds tail items for the super patterns. Finally, the algorithm checks whether the pattern being processed is in MHPS or not by the function Check_and_Add.

Note that the proof of the completeness and correctness of the MHP algorithm is presented in the Appendix.

4.2 An Example to Illustrate the MHP Algorithm

In this subsection, we describe the process of the MHP algorithm using a small sample dataset as shown in Table 2. Figure 2 highlights the whole process. As shown in the Figure, Initial Function first generates all the size-1 hyperclique patterns. Only item 10 will be pruned, since support(10)=0.1< θ ($\theta = 0.15$). Also, all these size-1 patterns are sorted by their support and we have $HP_1 = \{\{1\},\{2\},\{5\},\{6\},\{11\},\{3\},\{4\},\{7\},\{8\},\{9\}\}$.

In the BFS phase, Generate_and_Prune_Super Function generates size-2 candidate pattern set CP_2 from size-1 hyperclique pattern set HP_1. The super pattern of $\{1\}$ are first generated. $\{1,2\}$, $\{1,5\}$,$\{1,6\}$ are added into CP_2. The rest super pattern of $\{1\}$ are pruned by H-confidence Pruning. Also, since support({5,6} = support({5,11}) = support({5}) = 0.3. Items 6 and 11 are both PPE item of pattern {5}. The Equivalence Pruning Method will absorb 6 and 11 into the equivalence set of {5}, and update h-confidence value of {5} to $\frac{support(\{5,11\})}{support(\{11\})} = \frac{0.3}{0.4} = 0.75$. The rest super patterns of {5} are pruned by H-confidence Pruning. In a similar fashion, all the size-2 super pattern are generated and inserted into CP_2 since all of them are hyperclique patterns. Now CP_2 is: {{1,2}, {1,5}, {1,6}, {2,5}, {2,6}, {3,4}, {3,7}, {3,8}, {3,9}, {4,7}, {4,8}, {4,9}, {7,8}, {7,9}, {8,9}}.

In the second step of Generate_and_Prune_Super, patterns in CP_2 will succeed PPE items from their size-1 sub patterns. Item 6 will be added into equivalence sets of {1,5} and {2,5}, since it is in {5}.equivalence. Now we have HP_2 from CP_2: {{1,2}, {1,5(6)}, {1,6}, {2,5(6)}, {2,6}, {3,4}, {3,7}, {3,8}, {3,9}, {4,7}, {4,8}, {4,9}, {7,8}, {7,9}, {8,9}} (The item in the parenthesis are PPE items in equivalence item set). After generating HP_2, we can extract size-1 candidate maximal hyperclique pattern from HP_1. {5(6,11)}, {6(11)} and {11} do not have super union in HP_2. So $CMHP_1$ = {{5(6,11)}, {6(11)}, {11}}. Next, Function Check_and_Add will check whether the patterns in $CMHP_1$ has a super union in the current Maximal Hyperclique Pattern Set. Initially, MHPS is an empty set. The pattern {5,6,11}, the union of {5(6,11)}, will be first inserted into MHPS and MHPS = {{5,6,11}}. Since {5,6,11} is a super union of {6(11)}, the union of {6(11)} or {11} will not be inserted into MHPS.

In the second level loop, Function Generate_and_Prune_Super will generate CP_3 from HP_2. {1,2} is the first pattern in HP_2. By Prevalence Pruning, only {1,2,5} and {1,2,6} can be generated. Since support({1,2,5}) = support({1,2}) = 0.2, and item 5 is PPE item of {1,2}, 5 is absorbed into {1,2}.equivalence. Meanwhile, item 6 appears in {1,5}.equivalence and also a PPE item of {1,2}, so this item is transferred into {1,5}.equivalence. {1,2,6} will not be generated since 6 has already been in equivalence set of {1,2} now. The Function will not generate super candidate patterns for {1,5(6)}, {1,6}, {2,5(6)} and {2,6}, because any super pattern of them has at least one subpattern not contained in HP_2. For the rest patterns in HP_2, {3,4,7}, {3,4,8}, {3,4,9}, {3,7,8}, {3,7,9}, {3,8,9}, {4,7,8}, {4,7,9}, {4,8,9} and {7,8,9} will be generated and inserted into CP_3, since they are hyperclique patterns.

In the second step of Generate_and_Prune_Super, the size-3 candidate patterns will succeed PE items. {1,2(5,6)} is the only pattern in HP_2 which does not have super union in HP_3. So $CMHP_2$ = {{1,2(5,6)}}. Function Check_and_Add finds this pattern which does not have super union in current MHPS, and adds {1,2,5,6} into MHPS. Now, MHPS is {{5,6,11}, {1,2,5,6}}.

In the DFS phase, function Append_Tail will append tail items to the size-3 hyperclique pattern in HP_3. For instance, item 8 could be appended to {3,4,7}.tail since all the size-3 sub patterns of {3,4,7,8} appear in HP_3. After this step, HP_3 become: {{3,4,7-8,9}, {3,4,8-9}, {3,4,9}, {3,7,8-9}, {3,7,9}, {3,8,9}, {4,7,8-9}, {4,7,9}, {4,8,9}, {7,8,9}} (The items following '-' symbol are tail items). The recursive function Extract_MHP will generate Maximal Hyperclique Pattern from HP_3 in lexicographic order. First, this function will generate {3,4,7,8} and {3,4,7,9}. A dynamic Reordering Method compares their support and order these patterns by their support. Here they have same support and {3,4,7,8} is lexicographic before {3,4,7,9}, so we process {3,4,7,8} first. Item 9 will be appended into {3,4,7,8}.tail and {3,4,7,8,9} is generated.

Finally, Function Check_and_Add finds that the union of this pattern is not a sub pattern of any MHPS pattern and add the union into MHPS. Now MHPS = {5,6,11},{1,2,5,6},{3,4,7,8,9}}. Since {3,4,7,8,9} is in the leftmost path of {3,4,7}, all the candidate super patterns derived from {3,4,7} will be pruned. All other patterns in HP_3 will be pruned by Full Pruning since the unions of all of their itemset, equivalence set and tail set are sub patterns of {3,4,7,8,9}, which is in the current MHPS. Therefore, we have MHPS = {{5, 6, 11}, {1,2,5,6}, {3,4,7,8,9}} for θ=0.15 and H_c=0.55.

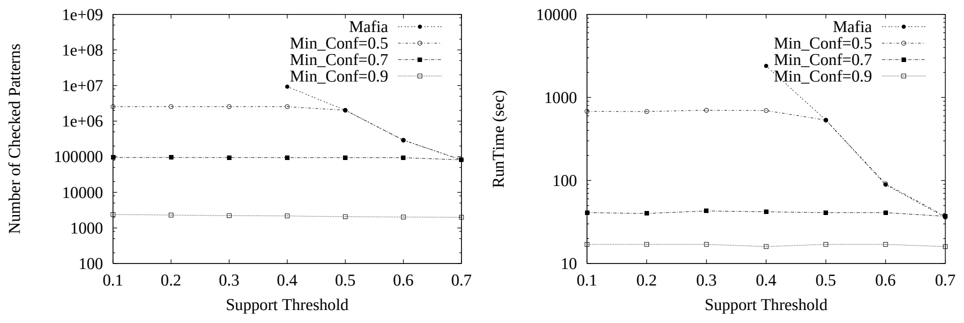
5 Experimental Evaluation

In this section, we present experiments to (1) evaluate the performance of the MHP algorithm, (2) analyze the effect of the equivalent pruning method in the BFS phase, (3) compare maximal hyperclique patterns to hyperclique patterns as well as maximal frequent patterns, and (4) show the application of maximal hyperclique patterns for identifying protein functional modules.

5.1 The Experimental Setup

Experiment Data Sets. Our experiments were performed on some real-world date sets, which are benchmark data sets for evaluating pattern mining algorithms. First, *pumsb* and *pumsb*∗ data sets[1] correspond to binary versions of a census data set. The difference between them is that pumsb* does not contain items with support greater than 80%. The $LA1$ data set is part of the TREC-5 collection[2] and contains news articles from the Los Angeles Times. In addition, the TAP-MS data set [8] is a protein complex data set, which summarizes large-scale experimental studies of multi-protein complexes for

[1] These two data sets are obtained from IBM Almaden research center at http://www.almaden.ibm.com/cs/quest/demos.html.
[2] The data set is available at http://trec.nist.gov.



(a) The number of patterns. (b) The Running Time

Fig. 4. Performance Comparisons on the Pumsb Data Set

the yeast *Saccharomyces Cerevisiae*. Some characteristics of these data sets are described in Table 3[3].

Table 3
Characteristics of Real-world Data Sets.

Data set	#Item	#Transaction	Source
Pumsb	2113	49046	IBM Almaden
Pumsb*	2089	49046	IBM Almaden
LA1	29704	3204	TREC-5
TAP-MS	1440	232	Gavin's Protein Complexes

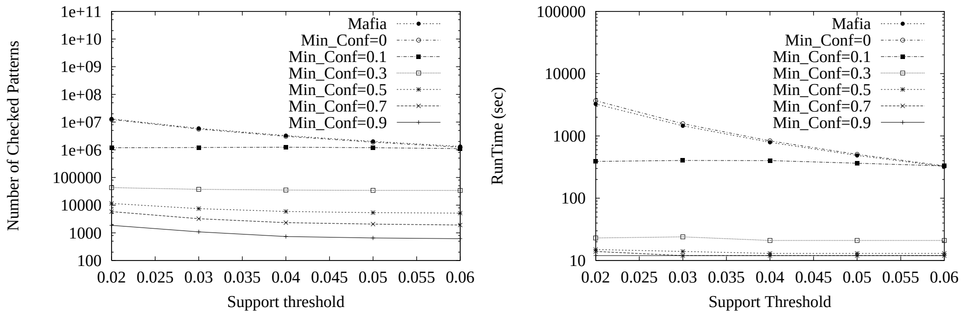
A Benchmark Algorithm. Recently, the MAFIA algorithm [6] was proposed to efficiently discover maximal frequent patterns. MAFIA is a pure DFS searching algorithm. As described in their paper, MAFIA can be several orders faster than some alternative methods, such as DepthProject, for mining maximal frequent patterns. The code of the DFS phase of the MHP algorithm is built on top of MAFIA, while we have added some new optimal methods. In this paper, we chose MAFIA as the base line for our performance evaluation.

The Experimental Platform. We implemented the MHP algorithms using C++ and all experiments were performed on a Pentium III 550MHz PC with 128 megabytes main memory, running Linux Redhat 6.1 operating system.

5.2 A Performance Comparison

Figure 4(a) illustrates the number of patterns that MHP and MAFIA generated at the different support and h-confidence thresholds on the pumsb data

[3] We have removed all the items which have not appeared in any transaction



(a) The number of patterns. (b) The Running Time

Fig. 5. Performance Comparisons on the Pumsb* Data Set

set. Note that, for all the experiments illustrated in this section, the level of BFS phase is 2. As can be seen, for MHP, the number of patterns generated is increasing with the decrease of the support threshold. In practice, the cross-support patterns, which contain items with different support levels, are weakly-related and are not desirable for real-world applications[18]. When the support threshold is low, MAFIA will generate too many cross-support patterns. However, the number of patterns generated by MHP can be significantly smaller than that of MAFIA even if a low h-confidence threshold is specified, since many cross-support patterns have been removed by MHP. Considering that, after generating a pattern, the algorithm needs to count support for the pattern. Support counting is the most time-consuming task during the pattern mining process, since the algorithm needs to retrieve all the transactions which include one of its sub-patterns, or for MAFIA, retrieve all the bits of the bitmap of this pattern [6]. Therefore, an algorithm is more efficient if a smaller number of patterns need to be generated.

The running time of MHP and MAFIA on the Pumsb data set is described in Figure 4 (b). In the figure, we can observe that the running time of MHP can be significantly reduced with the increase of h-confidence thresholds. Also, the running time of MHP can be much less than that of MAFIA even if we just set a little higher h-confidence threshold. The major reason is that the number of generated patterns of MHP is significantly smaller than that of MAFIA, and MHP doesn't need waste space to store the spacious patterns.

In addition, MAFIA is unable to generate patterns when the support threshold is less than or equal to 0.4, as it runs out of memory. Hui et al. have shown that nearly 96.6 percent of the items have supports less than 0.4[17]. MAFIA will fail to generate useful associations from the less popular items. In contrast, MHP algorithm can identify strong relations from these items. MHP can identify maximal hyperclique patterns when the support threshold is 0.1, if we set the h-confidence threshold to 0.5. In other words, MHP has the ability to identify patterns which can be difficult to identify for MAFIA. Hence, MHP can better explore the pattern space and find interesting patterns at low levels of support.

Similar results are also obtained from the pumsb* data set, as described in Figure 5. Since pumsb* removes all the popular items which have supports more than 0.8, the spacious patterns with popular items will not be generated. So MAFIA can find patterns when the support threshold is 0.02. There are still too many spacious patterns. For the pumsb* data set, the number of generated patterns of MHP is much smaller than that of MAFIA. And the running time of MHP can be several orders of magnitude less than that of MAFIA, even when we just set the h-confidence threshold as low as 0.3.

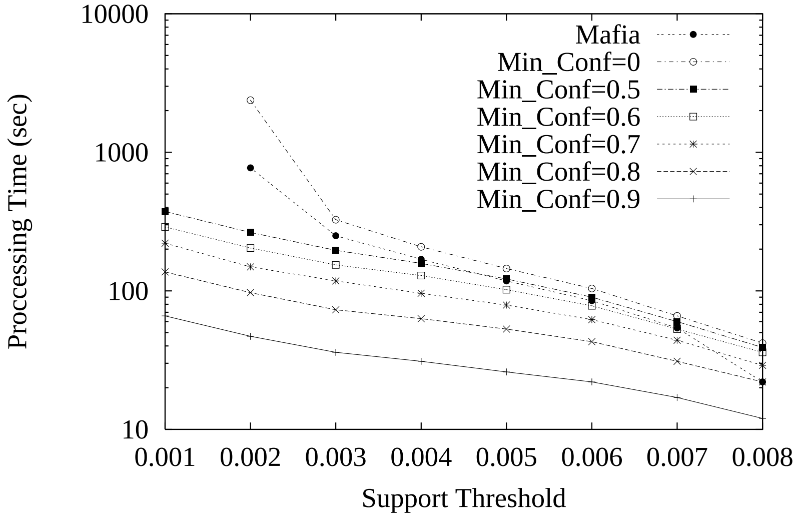
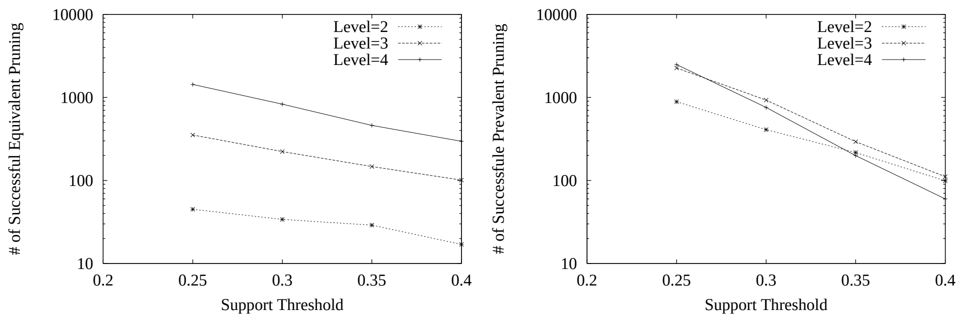


Fig. 6. Performance Comparisons on the LA1 Data Set

For the very sparse data set LA1, the hybrid algorithm is not as efficient as the MAFIA algorithm in mining frequent patterns. The processing time and generated time is demonstrated in Figure 6. MHP algorithm is about 4 times lower than the MAFIA when the support level is 0.002. The main reason is that we need spend many memories to store the size-2 patterns. There are 2,317 size-1 patterns and 59,603 size-2 patterns generated in BFS phase of MHP, so we need store 61,920 patterns. Every pattern needs about 2317/8 = 290 bytes for the items and 3204/8 = 400 bytes for the support information. Our MHP algorithm needs at least (290 + 400) * 61,920 = 42.7M bytes space to store them. The total main memory in our machine is only 128M, and there is about 82.3M left for the users. Since our approach also need many memory space during the processing procedure, we can't avoid data swapping between the main memory and the hard disk, which will greatly affect the speed. On the other side, MAFIA only need record 2317 F1 patterns and save more memory space. When we set the support threshold to 0.003, the number of size-2 pattern reduce to 8894, and the processing times of MHP and MAFIA are almost the same. Same as the pumsb and pumsb* data set, when we set



(a) Number of Equivalent Pruning in BFS phase

(b) Number of Partial Prevalent Pruning in DFS phase

Fig. 7. Effect of Different BFS Level, on the Pumsb* Data Set

a proper h-confidence threshold, MHP approach can be orders faster than the MAFIA and mine much lower support patterns.

5.3 The Effect of the Choices of Different Levels in the BFS Phase

In this subsection, we evaluate the effect of the choices of different search levels in the BFS phase. Indeed, if the search depth is deeper, we could get more equivalent pruning in the BFS phase. Since we get longer patterns in the first phase, we could prune more patterns with the partial prevalent pruning method in the second phase. However, this may result in more memory requirement. There is a tradeoff between memory usage and better pruning.

Figure 7(a) illustrates the number of equivalent pruning at different search levels in the BFS phase. As it can be seen, with the increase of search levels, the hybrid approach can prune 4-6 times more patterns.

Figure 7(b) shows the partial prevalent pruning in the DFS phase when the BFS levels are different in pumsb* data set. We observe that the approach with 3 levels in BFS can achieve partially prevalent pruning twice better than the approach with 2 levels. Also, the approach with 4 levels prunes much less than the approach with 3 levels. The main reason is that many size-4 patterns are pruned by prevalent pruning in BFS phase.

With the above experiments, it shows that the hybrid approach with 3 levels in the BFS phase may be better than the approaches with 2 or 4 levels.

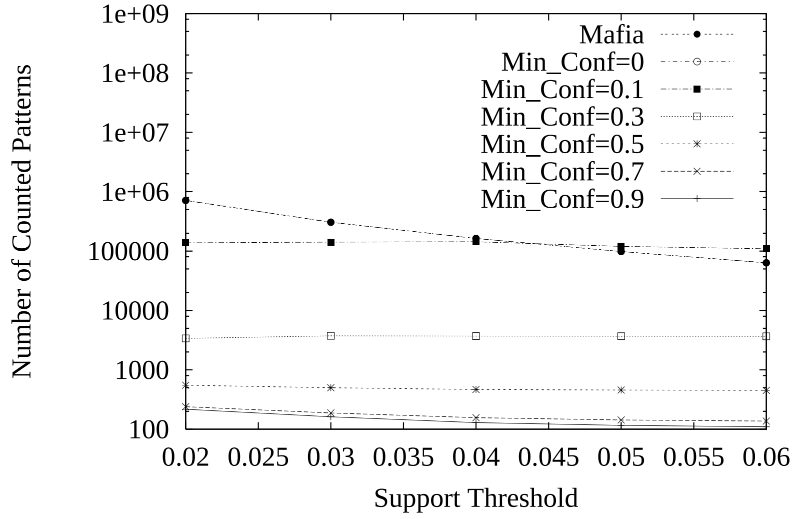


Fig. 8. The Number of MFI/MHP Patterns in the Pumsb* Data Set.

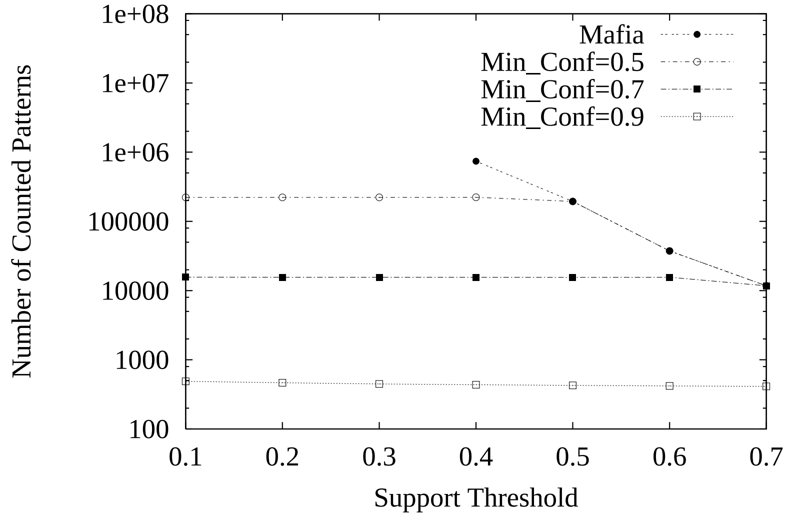


Fig. 9. The number of MFI/MHP Patterns in the Pumsb Data Set.

5.4 Maximal Hyperclique Patterns versus Maximal Frequent Patterns

Figure 8 and Figure 9 illustrate the number of maximal patterns identified by MHP and MAFIA on Pumsb* and Pumsb data sets respectively. As it can be seen, the number of maximal hyperclique patterns identified by MHP can be orders of magnitude smaller than the number of maximal frequent patterns identified by MAFIA. In other words, the number of maximal hyperclique patterns is much easier to manage than that of maximal frequent patterns. Indeed, in real-world applications, it is difficult to interpret several million maximal frequent patterns. However, it is possible to interpret the results of maximal hyperclique pattern mining.

5.5 *Maximal Hyperclique Patterns versus Hyperclique Patterns*

θ/H_c	0.99	0.95	0.90	0.85
0	149	503	4386	29744
0.2	90	441	4318	29671
0.4	25	375	3682	27507
0.5	21	360	3656	27466

(a) Number of HP Patterns

θ/H_c	0.99	0.95	0.90	0.85
0	70	149	641	2243
0.2	18	95	578	2175
0.4	25	91	569	2163
0.5	10	84	564	2154

(b) Number of MHP Patterns

Table 4
The number of maximal hyperclique patterns and hyperclique patterns generated on the pumsb data set.

Maximal hyperclique patterns correspond to a more compact representation of hyperclique patterns, while maximal hyperclique patterns may lose the information about support and h-confidence of their subsets. However, in some application domains, maximal hyperclique patterns provide sufficient information in terms of practical use, such as the use of maximal hyperclique patterns for pattern preserving clustering [17].

Table 4 illustrate the number of MHP patterns and HP patterns generated on the pumsb data set [4] . With the increase of the support threshold, the number of MHP patterns and HP patterns increase very slowly. In contrast, with the decrease of h-confidence thresholds, the number of HP patterns increases much faster than the MHP patterns. When the h-confidence is 0.85, quite low for some applications, the number of MHP patterns is 10 times smaller than the HP patterns. This indicates that the number of maximal hyperclique patterns is more manipulated than the number of hyperclique patterns.

5.6 *An Application of Maximal Hyperclique Patterns for Identifying Protein Functional Modules*

In this subsection, we describe an application of maximal hyperclique patterns for identifying protein functional modules - groups of proteins involved in common elementary biological function [16].

Figure 10 shows the subgraphs of the Gene Ontology (www.geneontology.org) corresponding to a maximal hyperclique pattern {`Cus1, Msl1, Prp3, Prp9, Sme1, Smx2, Smx3, Yhc1`} identified from the TAP-MS protein complex data.

[4] We only compare the patterns with size greater than 1.

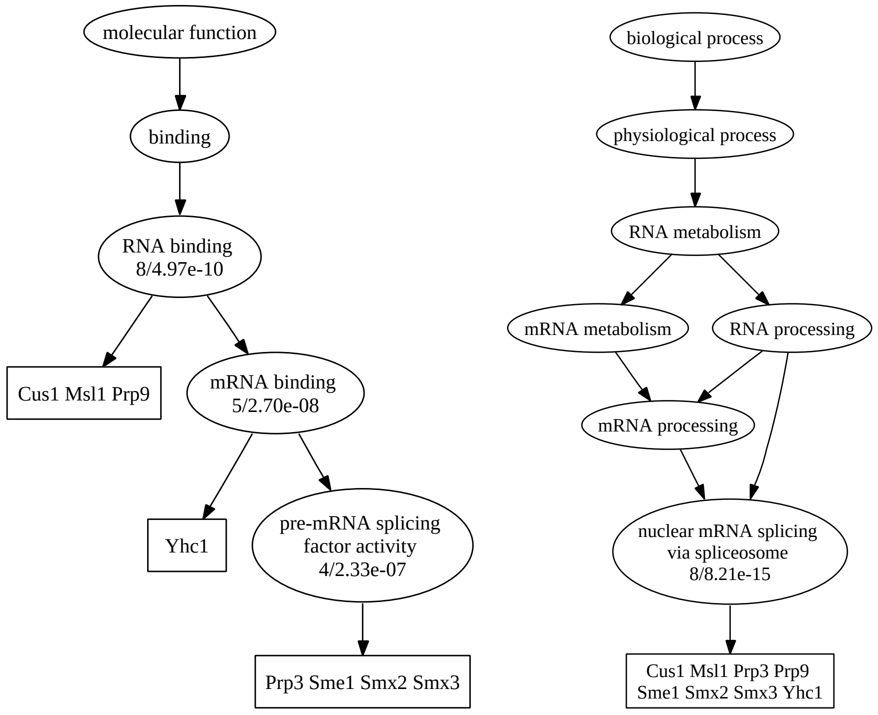


Fig. 10. The Gene Ontology annotations of pattern {Cus1, Msl1, Prp3, Prp9, Sme1, Smx2, Smx3, Yhc1}. Figure on the left shows subgraph of function annotation of the pattern. Figure on the right shows subgraph of process annotation. Proteins are listed in square box. Significant nodes are labeled with the number of proteins annotated directly or indirectly to that term and the p-value for the term.

The left subgraph in the figure is the molecular function annotation of the proteins in the pattern. Note that all 8 proteins from this pattern are annotated to the term *RNA binding* with p-value 4.97e-10. The p-value is calculated as the probability that n or more proteins would be assigned to that term if proteins from the entire genome are randomly assigned to that pattern. The smaller the p-value, the more significant the annotation. Among the pattern, 4 proteins {`Prp3, Sme1, Smx2, Smx3`} are annotated to a more specific term *pre-mRNA splicing factor activity* with p-value 2.33e-07. The annotation of these proteins confirms that each pattern form a module performing specific function. The right subgraph in Figure 10 shows the biological process this pattern is involved in. The proteins are annotated to the term *nuclear mRNA splicing via spliceosome* with p-value 8.21e-15 which is statistically significant.

6 Conclusions and Future Work

In this paper, we designed a two-phase Maximal Hyperclique Pattern (MHP) mining algorithm for finding maximal hyperclique patterns. This algorithm combines best features of both the Breadth First Search (BFS) and Depth

First Search (DFS) strategies. More specifically, we adapted DFS pruning methods, such as equivalence pruning, to a BFS approach and designed a hybrid search strategy for efficiently identifying maximal hyperclique patterns. In addition, we proved the correctness and completeness of the MHP algorithm. Finally, our experimental results on real-world data sets show that the MHP algorithm can be orders of magnitude faster than standard maximal frequent pattern mining algorithms and has the ability to identify patterns at extremely low levels of support even for dense data sets.

There are several directions for future work. First, there are some other optimal methods for pattern finding, such as tree projection, FP-tree, and diffset [1, 19, 10]. We plan to adapt some of these techniques into our algorithm. Also, since the concept of closed patterns is very desirable for some application domains, we would like to design algorithms for mining closed hyperclique patterns.

Acknowledgments

This work was partially supported by NSF grant # ACI-0305567. The content of this work does not necessarily reflect the position or policy of the government and no official endorsement should be inferred.

References

[1] Ramesh C. Agarwal, Charu C. Aggarwal, and V. V. V. Prasad. A tree projection algorithm for generation of frequent itemsets. *Journal of Parallel and Distributed Computing*, 61:350–371, 2001.

[2] Rakesh Agrawal, Tomasz Imielinski, and Arun N. Swami. Mining association rules between sets of items in large databases. In *Proc. 12th ACM SIGMOD International Conference on Management of Data, Washington, DC, USA*, pages 207–216, 1993.

[3] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules in large databases. In *Proc. 20th International Conference on Very Large Data Bases, Santiago de Chile, Chile*, pages 487–499, 1994.

[4] Roberto J. Bayardo. Efficiently mining long patterns from databases. In *Proc. 17th ACM SIGMOD International Conference on Management of Data, Seattle, Washington, USA*, pages 85–93, 1998.

[5] Roberto J. Bayardo and Rakesh Agrawal. Mining the most interesting rules. In *Proc. 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Diego, CA, USA*, pages 145–154, 1999.

[6] Douglas Burdick, Manuel Calimlim, and Johannes Gehrke. Mafia: A maximal frequent itemset algorithm for transactional databases. In *Proc. 17th IEEE International Conference on Data Engineering, Heidelberg, Germany*, pages 443–452, 2001.

[7] Edith Cohen, Mayur Datar, Shinji Fujiwara, Aristides Gionis, Piotr Indyk, Rajeev Motwani, Jeffrey D. Ullman, and Cheng Yang. Finding interesting associations without support pruning. In *Proc. 16th IEEE Conference on Data Engineering, San Diego, California, USA*, pages 489–499, 2000.

[8] AC Gavinet, M Bosche, R Krause, and M Marzioch et al. Functional organization of the yeast proteome by systematic analysis of protein complexes. *Nature*, 415:141–147, 2002.

[9] Gösta Grahne, Laks V. S. Lakshmanan, and Xiaohong Wang. Efficient mining of constrained correlated sets. In *Proc. 16th IEEE Conference on Data Engineering, San Diego, California, USA*, pages 512–521, 2000.

[10] Jiawei Han, Jian Pei, and Yiwen Yin. Mining frequent patterns without candidate generation. In *Proc. 19th ACM SIGMOD International Conference on Management of Data, Dallas, TX, USA*, pages 1–12, 2000.

[11] Xuemin Lin, Yijun Li, and Chi Ping Tsang. Applying on-line bitmap indexing to reduce counting costs in mining association rules. *Information Sciences*, 120:197–208, 1999.

[12] Bing Liu, Wynne Hsu, and Yiming Ma. Mining association rules with multiple minimum supports. In *Proc. 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Diego, CA, USA*, pages 337–341, 1999.

[13] Ron Rymon. Search through systematic set enumeration. In *Proc. 3rd International Conference on Principles of Knowledge Representation and Reasoning, Cambridge, Massachusetts, USA*, pages 539–550, 1992.

[14] Li Shen, Hong Shen, and Ling Cheng. New algorithms for efficient mining of association rules. *Information Sciences*, 118:251–268, 1999.

[15] Ke Wang, Yu He, David Wai-Lok Cheung, and Francis Y. L. Chin. Mining confident rules without support requirement. In *Proc. 10th ACM International Conference on Information and Knowledge Management, Atlanta, Georgia, USA*, pages 89–96, 2001.

[16] Hui Xiong, X. He, Chris Ding, Ya Zhang, Vipin Kumar, and Stephen R. Holbrook. Identification of functional modules in protein complexes via hyperclique pattern discovery. In *Proc 10th the Pacific Symposium on Biocomputing, Big Island, Hawaii, USA*, pages 221–232, 2005.

[17] Hui Xiong, Michael Steinbach, Pang-Ning Tan, and Vipin Kumar. Hicap: Hierarchical clustering with pattern preservation. In *Proc. 4th SIAM International Conference on Data Mining (SDM), Orlando, Florida, USA*, pages 279–290, 2004.

[18] Hui Xiong, Pang-Ning Tan, and Vipin Kumar. Mining strong affinity association patterns in data sets with skewed support distribution. In *Proc. 3rd IEEE International Conference on Data Mining, Melbourne,*

Florida, USA, pages 387–394, 2003.

[19] Mohammed Javeed Zaki. Scalable algorithms for association mining. *IEEE Transaction on Knowledge and Data Engineer*, 12:372–390, May-June 2000.

[20] Mohammed Javeed Zaki and Ching-Jiu Hsiao. Charm: An efficient algorithm for closed itemset mining. In *Proc. 2nd SIAM International Conference on Data Mining (SDM), Arlington, VA, USA*, pages 457–473, 2002.

Appendix: A Proof of the Completeness and Correctness of the MHP Algorithm

Lemma 2 *If a hyperclique pattern, HP_1, is generated in the BFS phase, none of the items in HP_1.item could be a PPE item of any sub pattern of HP_1.*

Proof: We prove this lemma by contradiction. Suppose HP_k.item=$\{i_1, i_2, ..., i_k\}$, and i_l is a PPE item of HP_m, where HP_m is a sub pattern of $HP_k, l \leq k$. We have $HP_m \subseteq$ of $\{i_1, i_2, ..., i_{l-1}\}$, and i_l should also be a PPE item of this size-(l-1) pattern. According to our algorithm, $\{i_1, i_2, ..., i_{l-1}\}$ will absorb i_l into the equivalence set. So the size-l pattern, $\{i_1, i_2, ..., i_l\}$, will not be generated. HP_k cannot be generated either. The lemma is proved. □

Lemma 3 *When a hyperclique pattern, HP_1, is generated in the BFS phase and the size of $HP_1 < L$, if $\exists$ an item i, (1) all the items in the HP_1.item are lexicographic before i, (2)i is not an equivalence item of HP_1, and (3)HP_1.item $\cup$ $\{i\}$ is also a hyperclique pattern, then HP_1.item $\cup$ $\{i\}$ will be generated by the MHP algorithm.*

Proof:

Suppose HP_1 is a minimal size pattern in this kind of ungenerated patterns, where HP_1.item=$\{i_1, i_2, ..., i_k, i\}$, and the sub pattern HP' which *itemset* is $\{i_1, i_2, ..., i_k\}$ has been generated. Then, all the size-(k-1) sub pattern of HP' should have been generated, and i satisfies all the three conditions in this lemma to them. All the size-k sub patterns of HP_1 will be generated since they are smaller than HP_1. HP_1 will been generated in the next level apriori-gen. This leads to contradictory. □

Lemma 4 *If a hyperclique pattern, HP_1, is generated in the BFS phase, all PPE items of its sub patterns will be added to the HP_1.equivalence by the MHP algorithm.*

Proof: Suppose HP_1 is the minimal such hyperclique pattern which breaks this lemma, and i is any PPE item of one of HP_1's sub pattern, HP_1.item =

$\{i_1, i_2, ..., i_k\}$.

If i is also a PPE item of HP_1's pure sub pattern, i will appear in the equivalence set of this sub pattern. HP_1 could succeed this PE item from this pattern directly or indirectly.

If i is not a PPE item for any pure sub patterns, it should be a PPE item of HP_1. Now $i_k < i$ and i is not an equivalence item of any pure sub patterns of HP_1. By Lemma 3, all the size-k sub patterns of $\{i_1, i_2, ..., i_k, i\}$ will be generated. The item i will be absorbed or transferred to HP_1.equivalence in the next level apriori-gen. Here, we observe a phenomenon: if i is transferred to the equivalence set, it will also be added to the equivalence set with absorbing and succeeding methods if the transferring methods is not applied. $\Box$

Lemma 5 *An equivalence item, which is transferred by a hyperclique pattern HP_1, could also be added to equivalence set with the absorbing or succeeding method if the transferring method is not applied.*

Proof: If an item is transferred from another pattern, this item is also a PPE item of one of its sub pattern. In the proof of Lemma 4, we find this item will be added to the equivalence set of this sub pattern even without transferring method. It will also be added to the equivalence set of this pattern. $\Box$

Lemma 6 *For a hyperclique pattern HP, all items in HP.equivalence are PPE items of some sub pattern of the pattern HP.*

Proof: This lemma is correct, since all the three methods, absorbing, succeeding and transferring, add an item into HP_1.equivalence only when this item is a PPE item of one of HP_1's sub pattern. $\Box$

Lemma 7 *For a hyperclique pattern, HP, if an item is a PPE item of a sub pattern of HP.union, it is also a PPE item of a sub pattern of HP.item; similarly, if an item is a PPE item of a sub pattern of HP.item, it is also a PPE item of a sub pattern of HP.union.*

Proof: If an item i is a PPE item of a sub pattern, SP, of HP.union, we could construct a pattern $HP_1.item = \{i' | i' \in HP.union \wedge i' < i\}$. We have $HP_1 \supseteq SP$ and i is a PPE item of HP_1. $\forall$ item i'', where $i'' \in (HP_1.item \bigcap HP.equivelance)$, i'' is a PPE item of a sub pattern of HP, by the Lemma 6. Since this sub pattern should also be a subset of HP_1, i'' is a PE item of $HP_1.item/i''$. So i is also a PPE item of $HP_1.item/i''$. After getting rid of all the items in HP.equivalence, we could find that i is a PPE item of ($HP_1.item \bigcap HP$.item), which is a sub pattern of HP.item.

Since HP.union is a super set of HP.item, the second part can be proved in a similar fashion $\Box$.

Definition 16 *For a hyperclique pattern P_1, if (1) P_2 is a hyperclique pattern, (2) P_2.item is a subset of P_1.item, and (3) the union of P_2 is a super set of P_1.item, P_2 is a* **Covering Pattern** *of P_1.*
We have support(P_2.item)=support(P_1.item)=support(P_2.union).

Lemma 8 *If a pattern is a hyperclique pattern, one of its covering patterns must be generated by our approach if the full pruning and leftmost pruning methods are not applied. If these two method are applied, then one of the covering patterns of its super pattern must be generated.*

Proof: We first prove that without the full pruning and leftmost pruning methods, a covering pattern of the hyperclique pattern will be generated. We prove it with contradiction. Suppose HP_1 is a minimal such hyperclique pattern without generated covering pattern. HP_1.item=$\{i_1, i_2, ..., i_k\}$. Let HP'.item = $\{i_1, i_2, ..., i_{k-1}\}$. HP' should have a Covering Pattern(CP_1) generated by our algorithm. There are two cases for generating CP_1:

The first case is that CP_1 is generated in BFS phase. (1)If i_k is an equivalence item of any sub pattern of CP_1, i_k should be also a PPE item of this sub pattern. CP_1 will add i_k into its equivalence itemset and become a covering pattern of HP_1. (2)If i_k is not an equivalence item for any sub pattern of CP_1, by the Lemma 3, a new pattern $CP_1.item \cup \{i_k\}$ will be generated. This pattern will be the Covering Pattern of HP_1.

The other case is that CP_1.item is generated in the DFS phase. (1) If i_k is an equivalence item of any size-L sub patterns of CP_1, CP_1 will succeed i_k and become the covering pattern of HP_1. (2)If i_k is not an equivalence item for any size-L sub pattern of CP_1, by the Lemma 3, all the size-L sub patterns of $CP_1.item \cup \{i_k\}$ will be generated. In the Append_Tail Function, the size-L sub patterns of CP_1 will add i_k into their tail sets. DFS phase could not get rid of i_k from the tail sets of CP_1's sub pattern's unless adding it to the equivalence sets. If i_k is an equivalence item of CP_1, it will appear in CP_1.equivalence, otherwise, the pattern CP_1.item $\cup$ $\{i_k\}$ will be generated while processing CP_1. Both of them will generate a Covering Pattern of HP_1.

From the above, both cases lead to contradiction.

With the Full Pruning and Leftmost Pruning methods, a node will be removed if the union of its itemset, equivalence set and tail set has a super pattern in current MHPS. But in this case, all the hyperclique patterns could be generated by this node will have a super pattern in MHPS, which means they should have a covering pattern of the super maximal pattern generated already. $\square$

Theorem 1 *The MHP algorithm is complete. In other words, all the Maximal Hyperclique Patterns will be identified by the MHP algorithm.*

Proof: $\forall$ maximal hyperclique pattern MHP_1, by the Lemma 8, one of its super pattern's covering pattern will be generated. Since MHP_1 is maximal, the union of this covering pattern should be MHP_1's itemset $\Longrightarrow$ MHP_1 can be identified by the MHP algorithm. $\Box$

Theorem 2 *The MHP algorithm is correct. In other words, any pattern identified by the MHP algorithm is a maximal hyperclique pattern.*

Proof: First, any pattern identified by the MHP algorithm is a hyperclique pattern. We only need to show the pattern is maximal.

Case 1: If we find a pattern has no super hyperclique pattern or none of its super hyperclique pattern's union set is superset of the pattern's union in DFS phase. According to the procedure of DFS searching, we definitely cannot find a super union in the rest. If there is no super pattern in current MHPS set, this union should be a MHP pattern.

Case 2: If we find a pattern without a super union in the next level pattern in BFS phase, we could also make sure that the union is a MHP if there is no super pattern in current MHPS set.

Assume that we generated a hyperclique pattern, HP_1, in BFS phase, where HP_1.item=$\{i_1, i_2, ..., i_k\}$ and HP_1.union is not a MHP pattern. There should exists at least one item, i, which belongs to HP.union's super maximal hyperclique pattern but not belongs to HP.union. By Lemma 4, i cannot be PPE item of any sub pattern of HP_1.

Since full pruning and leftmost pruning methods are not applied in the BFS phase, a covering pattern (CP_1) of $HP_1.item \cup \{i\}$ will be generated and the size of $CP_1 \leq$ k+1. By Lemma 6, we know $i \in CP_1$.item. $\forall$ item $i' \in HP_1$.equivalence, also by Lemma 6, i' is a PPE item of a sub pattern of HP_1. Since CP_1.union $\supset HP_1$.item. i' should also be a PPE item of a sub pattern of CP_1.union. By Lemma 7, i' is a PPE item of a sub pattern of CP_1 and is added into CP_1.equivalence. So, we know that CP_1.union is superset of HP_1.union.

(1) If size(CP_1) = k+1, HP_1.union will not be considered as a MHP pattern since there is a superset in the next level.

(2) If size(CP_1) = k, a item in HP_1.item will appear in CP_1.equivalence. Since this item should be a PPE item of a sub pattern of HP_1, by Lemma 2, i should in this sub pattern. So i is lexicographic before this item and CP_1 is lexicographic before HP_1. CP_1 will be processed before HP_1. If CP_1.union is added into MHPS, we will find there is already a super set of HP_1.union in MHPS while processing HP_1. If CP_1.union is not added into MHPS, a size k+1 pattern, whose union is a super set of CP_1.union will be generated. This pattern's union is also a super set of HP_1.union.

(3) If size(CP_1) $<$ k, similar to (2), CP_1's super patterns are also lexicographic before HP_1. There are two cases for the super patterns of CP_1, one is there is one super pattern's union are added into MHPS when its size $\leq$ k; the other case is that a size-(k+1) super pattern are generated. HP.union cannot be added into MHPS in both cases.

Now it is guaranteed that our algorithm will not identify any non-maximal hyperclique pattern as a MHP pattern. $\square$

Note that if we set the search depth in the BFS phase large enough, our algorithm becomes a pure BFS algorithm. Also, if we set the h-confidence threshold to zero, the algorithm will find the maximal frequent itemsets.