A Hybrid Approach for Mining Maximal Hyperclique Patterns

Yaochun Huang *
Computer Science
Univ. of Texas - Dallas
yxh038100@utdallas.edu

Hui Xiong
Computer Science
University of Minnesota
huix@cs.umn.edu

Weili Wu, Zhongnan Zhang
Computer Science
Univ. of Texas - Dallas
{*weiliwu, zxz021000*}*@utdallas.edu*

Abstract

A hyperclique pattern [12] is a new type of association pattern that contains items which are highly affiliated *with each other. More 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 a new algorithm for mining maximal hyperclique patterns, which are desirable for pattern-based clustering methods [11]. This 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. As demonstrated by our experimental results, the performance of our algorithm can be orders of magnitude faster than standard maximal frequent pattern mining algorithms, particularly at low levels of support.*

Keywords
Data Mining, H-confidence, Hyperclique Pattern

1. Introduction

The association-rule mining problem [3] is concerned with finding relationships among the items in a large-scale data set. In the past decade, association-rule mining has been the subject of extensive research in data mining [1, 2, 3, 4, 8]. Given a set of transactions, the objective of association-rule mining is to extract all rules of the form $X \Rightarrow Y$, where X and Y are sets of items, that 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 [12]. Instead, a hyperclique pattern [12] was proposed as a new type of association pattern that contains 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 are desirable for pattern preserving clustering [11], which can produce easily interpretable clustering results. However, to our best knowledge, there is no efficient algorithm 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 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 subsets is not a hyperclique pattern. This pruning is based on the anti-monotone property of support and h-confidence measures. The drawback 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, 9], can be applied. However,

[1]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.

the DFS strategy cannot apply the $Prevalence\ Pruning$ method, since we do not generate all subsets of a candidate pattern for this strategy.

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 will be 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, 9], we adapt the equivalent pruning method to 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.

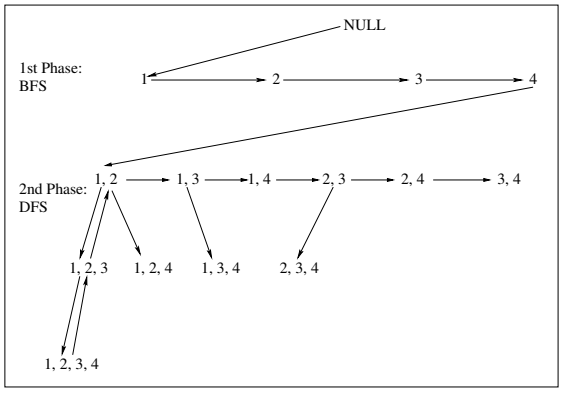

Figure 1. An Illustration of the Hybrid Mining.

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 classic 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.*

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'** $\notin$ **MHP**.

Definition 4 *The* **order** *of items: for two different item* 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*. We write it as* $i_1 < i_2$.

In the rest of this paper, we arrange the items in an itemset in order, except for special mention.

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 could be also written as* $P_1 < P_2$.

3. A Framework for Mining Maximal Hyperclique Patterns

In this section, we present a two-phase maximal hyperclique pattern mining framework. In the first BFS phase, we retrieve all the size-L hyperclique patterns. In other words, the first L levels of the $lexicographic\ tree$ [10] 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).

3.1. Basic Definition

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 [10, 1]. All tail items are included in P.tail.

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.*

Table 1 shows a sample data set and Table 2 shows the support of items in the sample data set. Also, for this sample data set, Figure 2 illustrates the two-phase maximal hyperclique pattern mining process. In the figure, a node represents a pattern, P. As can be seen, in the upper layer, numbers represent the items in P.item and numbers in the braces are the items in P.equivalence. In addition, numbers following the short line represents the items in P.tail. There are two rows in the lower part of the figure: the first one shows the support of patterns and the second row shows the corresponding h-confidence of patterns.

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

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

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

If an item i is an equivalence item with a pattern P and $\frac{support(P)}{support(i)} < H_c$, a minimum h-confidence threshold, then the union of $\{i\}$ and P.item cannot be a hyperclique pattern.

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

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, 9]. However, the direct use of 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 an PE item for $\{1, 5\}$. In other words, item 11 cannot be added into $\{5\}$, since we do not know whether the super patterns of $\{5\}$ have item 11 as PE item or not. In addition, we apply the equivalence pruning method in the BFS phase. Consider that adding items may change the size of patterns, we need to maintain a set to store a pattern's PE items.

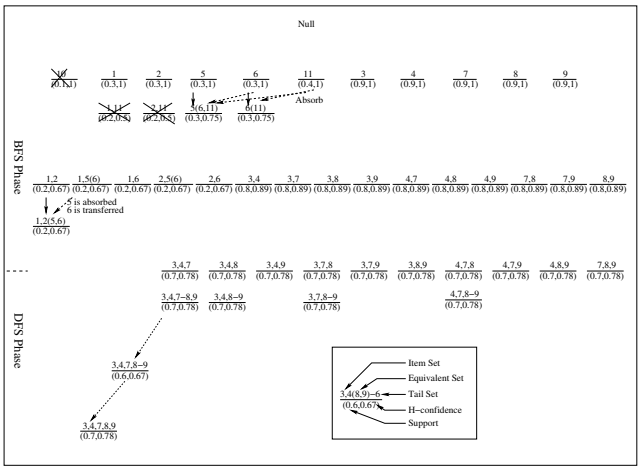

Figure 2. An Illustration of the Two-Phase Maximal Hyperclique Pattern Mining Process.

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

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 target pattern of P should be P.item $\cup$ P.equivalence.

Definition 11 Union *of a pattern, P.union, is $P.item \cup P.equivalence$. P.union is the target itemset of P. Indeed, support(P.union)=support(P.item).*

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

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.*

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.*

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 the items which have support 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 applies 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 the following.

Prevalence Pruning. In the apriori-like algorithm, a size-k pattern P_k ($P_k.item = \{i_1, i_2, ..., i_k\}$) 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. This is a standard pruning method [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}) [12]. 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. In Figure 2, support({5, 6})=support({5})=0.3, we add item 6 to {5}'s equivalence set and prune {5, 6}.

When generating a size-k hyperclique pattern HP_k, if the items in its size-(k-1) sub hyperclique patterns' equivalence sets are PE items of HP_k, HP_k can **succeed** these items to its own equevalence 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 limitatin 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 **tranfer** 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 the pattern {1, 5}.

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, 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 these patterns' tail sets. 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' the pattern's equivalence set. 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 in 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]. 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 MHPS, we can skip all the other paths of $\{3, 4, 7\}$, and continue to process the next pattern.

Dynamic Reordering. Bayardo showed that the benefit of dynamically reordering super patterns of HP is significant [5]. The mining speed will be 2 to 4 times faster. We sort the super patterns in the increasing order of support.

H-confidence Pruning. Similar to the BFS phase, for a tail item (i') of a hyperclique pattern(HP), if $\frac{hconf(HP)}{support(i')} < H_c$, 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 size-L sub pattern of HP is not generated, we need not generate it.

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.

4. The MHP Algorithm

Figure 3 shows an overview of the hybrid MHP algorithm for mining maximal hyperclique patterns. As can be seen, there are two phases: the BFS phase and the DFS phase in the algorithm.

4.1. Algorithm Description

In the first BFS phase, Initial Function generates the size-1 frequent patterns, which are also size-1 hyperclique patterns, and items are sorted in 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 second phase, the Append_Tail Function adds the size-L patterns' tail item. Extract_MHP is the major function for DFS mining. The traditinal optimal methods, full pruning, leftmost pruning, and equivalence pruning, and new methods, 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 add 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.

4.2. Completeness and Correctness

Here, we prove the completeness and correctness of our MHP algorithm. To facilitate our discussion, we first introduce some lemma and a new concept, **Covering Pattern**.

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

Proof: This lemma proof as well as some following lemma proofs are presented in our Technical Report [7]

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, HP_1.item $\cup$ {i'} will be generated by our algorithm.*

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

Lemma 5 *For an equivalence item which is transferred by a hyperclique pattern HP, it could also be added to equivalence set with absorbing or succeeding if we do not use transferring method.*

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

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. Obviously, support(P_2.item)=support(P_1.item)=support(P_2.union).*

MHS ALGORITHM

Input: (a) P = {A Pattern}
(b) $Data$ = {A DataSet represent a set of transaction }
(c) θ: 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 generate from P

Phase I: generate Hyperclique Patterns by BFS

1. HP_1= Initial(CP_1, θ, H_c, $Data$);
2. for (k=1;$k < L$;k++) do
3. CP_{k+1} = Generate_and_Prune_Super(HP_k, θ, 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, θ, 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 haven't a super union in P_{super}
14. Check_and_Add(P,MHPS);

Figure 3. Overview of the MHP Algorithm

Lemma 7 *If a pattern is a hyperclique pattern, one of its Covering Pattern must be generated.*

Lemma 8 *The MHP algorithm is complete. In other words, all the Maximal Hyperclique Patterns will be identified by the MHP algorithm.*

Lemma 9 *The MHP algorithm is correct. In other words, any patterns identified by the MHP algorithm are maximal hyperclique patterns.*

Note that if we set the search depth in the BFS phase large enough, our algorithm becomes a pure BFS algorithm. This means equivalence pruning will work correctly in an apriori-like algorithm for mining maximal hyperclique pattern. Additionally, if we set the h-confidence threshold to zero, the algorithm finds maximal frequent itemsets.

5. Experimental Evaluation

In this section, we present extensive experiments to evaluate the performance of the MHP algorithm. Specifically, we demonstrate: (1) a computation performance comparison between the MHP algorithm and standard maximal frequent pattern mining algorithms and (2) the effect of the MHP algorithm on finding maximal hyperclique patterns.

5.1. The Experimental Setup

Our experiments were performed on two real-world date sets $pumsb$ and $pumsb*$, which are benchmark dense data sets for evaluating pattern mining algorithms. These two data sets are obtained from IBM Almaden at http://www.almaden.ibm.com/cs/quest/demos.html. Recently, the MAFIA algorithm [6] was proposed to efficiently discover maximal frequent patterns. As shown in their paper, MAFIA can be several orders better than some alternatives, such as DepthProject, for mining maximal frequent patterns. Hence, we chose MAFIA as the base line for our performance evaluation. Finally, please note that only the size-2 patterns are generated in the first BFS phase.

Experimental Platform We implemented the MHP algorithms using C++ and all experiments were performed

on a Pentium III 550MHz PC machine with 128 megabytes main memory, running Linux Redhat 6.1 operating system.

5.2. A Performance Comparison

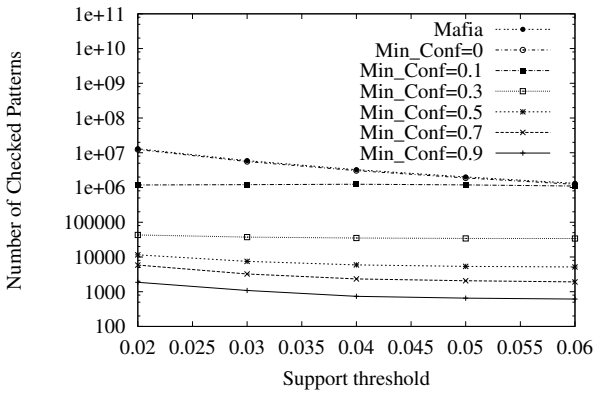

Figure 4. The Number of Checked Patterns on the Pumsb* Data Set

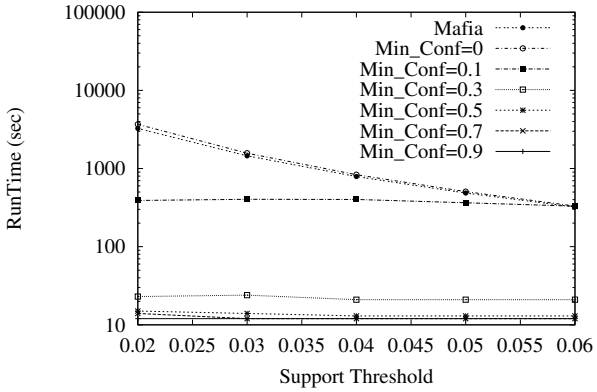

Figure 5. The RunTime Comparison on the Pumsb* Data Set

Figure 4 shows the number of patterns that MHP and MAFIA have to check during the pattern mining process on the pumsb* data set. As can be seen, for MHP, the number of checked patterns is increased with the decrease of the h-confidence threshold. However, the number of checked patterns of MHP can be significantly smaller than that of MAFIA even if a low h-confidence threshold is specified. To check a pattern, we need to count the support of the patterns. Counting the support of a pattern is the most time-consuming task during the pattern mining process, since we need to retrieve all the transactions which include one of its sub-pattern, or for Mafia, retrieve all the bit of the bitmap of this pattern [6]. Therefore, an algorithm is more efficient if smaller number of patterns need to be checked.

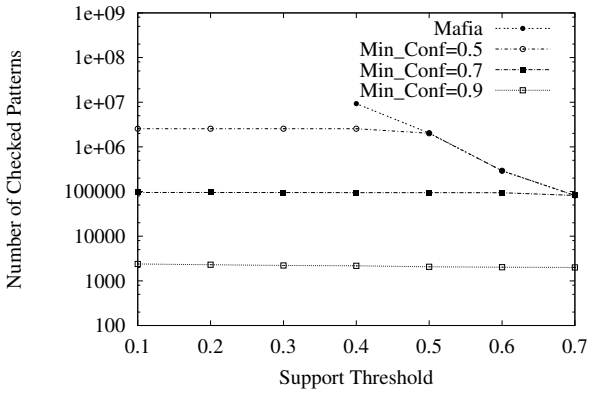

Figure 6. The Number of Checked Patterns on the Pumsb Data Set

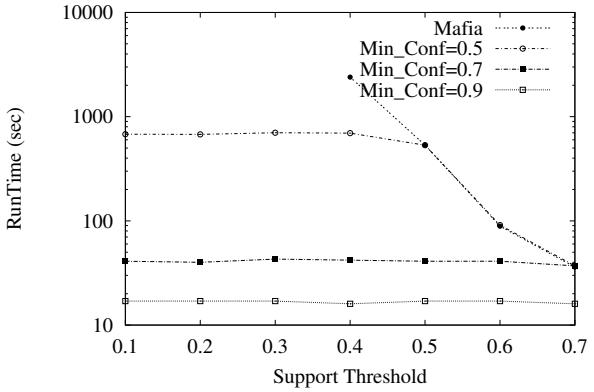

Figure 7. The RunTime Comparison on the Pumsb Data Set

The runtime comparison of MHP and MAFIA on the Pumsb* data set is shown in Figure 5. In the figure, we can observe that the runtime of MHP can be significantly reduced with the increase of h-confidence thresholds. Also, the runtime of MHP can be several orders of magnitude less than that of MAFIA even if the h-confidence threshold is as low as 0.3. The reason is that the number of checked patterns of MHP is signficantly smaller than that of MAFIA.

Similar results are also obtained from the pumsb data set, as shown in Figure 6 and Figure 7. For the pumsb data set, the number of checked patterns of MHP is much smaller than that of MAFIA and the runtime of MHP can be significantly less than that of MAFIA.

5.3. The Effect of the MHP Algorithm on Finding Maximal Hyperclique Patterns

Figure 8 and Figure 9 show the number of maximal patterns identified by MHP and MAFIA on Pumsb* and Pumsb data sets respectively. As 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

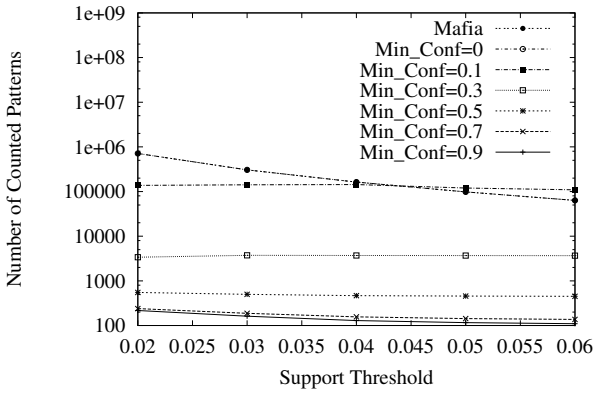

Figure 8. The Number of MFI/MHP Patterns in the Pumsb* Data Set.

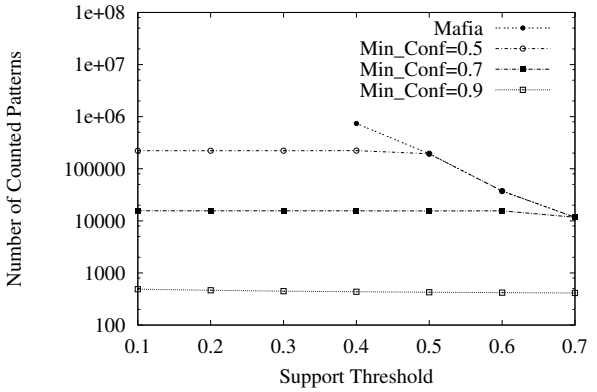

Figure 9. The number of MFI/MHP Patterns in the Pumsb Data Set.

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.

In addition, due to the memory limitation, we cannot extract maximal frequent patterns with MAFIA on the Pumsb data set if the support threshold is less than 0.4, as shown in Figure 7. In contrast, 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.

6. Conclusions and Future Work

In this paper, we present a two-phase Maximal Hyperclique Pattern (MHP) mining algorithm, which combines best features of both the BFS strategy and the DFS strategy. More specifically, we adapted DFS pruning methods, such as equivalence pruning, to an apriori-like approach. In addition, we proved the correctness and completeness of the MHP algorithm. Finally, our experimental results show that the MHP algorithm can be several orders of magnitude faster than standard maximal frequent pattern mining algorithms and has the ability to identify patterns at extremely low levels of support in dense data sets.

There are several directions for future work. First, in this paper, we only generate the size-2 patterns in the BFS phase. It will be interesting to investigate the impact on the performance if the first phase is stopped at a deeper level. Also, the projection is a very efficient method for finding patterns, especially for parallel implementation of pattern mining algorithms [1]. We plan to adapt the projection ideas into our algorithm and design an efficient parallel algorithm for mining maximal hyperclique patterns.

References

[1] R. Agarwal, C. Aggarwal, and V. Prasad. A Tree Projection Algorithm For Generation of Frequent Itemsets. pages 350–371, Feb 2001.

[2] R. Agrawal, T. Imielinski, and A. Swami. Mining Association Rules between Sets of Items in Large Databases. In *Proc. of the ACM SIGMOD Conference on Management of Data*, pages 207–216, May 1993.

[3] R. Agrawal and R. Srikant. Fast Algorithms for Mining Association Rules. In *Proc. of the 20th Int'l Conference on Very Large Data Bases*, 1994.

[4] R. Bayardo. Efficiently mining long patterns from databases. In *Proc. of the ACM SIGMOD Conference*, 1998.

[5] R. Bayardo and R. Agrawal. Mining the Most Interesting Rules. In *Proc. of the ACM SIGKDD Conference*, 1999.

[6] D. Burdick, M. Calimlim, and J. Gehrke. Mafia: A Maximal Frequent Itemset Algorithm for Transactional Databases. In *Proc. of IEEE Conf. on Data Engineering*, 2001.

[7] Y. Huang, H. Xiong, W. Wu, and Z. Zhang. A Hybrid Approach for Mining Maximal Hyperclique Patterns. In *In Technical Report UTDCS-34-04, Department of computer science, University of Texas - Dallas*, 2004.

[8] J.Han, J.Pei, and Y. Yin. Mining Frequent Patterns without Candidate Generation. In *Proc. of the ACM SIGMOD International Conference on Management of Data*, 2000.

[9] M.J.Zaki and C.Hsiao. ChARM: An efficient algorithm for closed itemset mining. In *Proc. of 2nd SIAM International Conference on Data Mining*, 2002.

[10] R.Rymon. Search through Systematic Set Enumeration. In *Proc. Third Int'l Conference on Principles of Knowledge Representation and Reasoning*, 1992.

[11] H. Xiong, M. Steinbach, P.-N. Tan, and V. Kumar. HICAP: Hierarchial Clustering with Pattern Preservation. In *Proc. of 2004 SIAM International Conference on Data Mining (SDM)*, pages 279–290, 2004.

[12] H. Xiong, P.-N. Tan, and V. Kumar. Mining Strong Affinity Association Patterns in Data Set with Skewed Support. In *Proc. of the Third IEEE International Conference on Data Mining (ICDM)*, 2003.