

Hyperclique pattern discovery

Hui Xiong · Pang-Ning Tan · Vipin Kumar

Received: 11 February 2004 / Accepted: 6 February 2006
© Springer Science + Business Media, LLC 2006

Abstract Existing algorithms for mining association patterns often rely on the support-based pruning strategy to prune a combinatorial search space. However, this strategy is not effective for discovering potentially interesting patterns at low levels of support. Also, it tends to generate too many spurious patterns involving items which are from different support levels and are poorly correlated. In this paper, we present a framework for mining highly-correlated association patterns called hyperclique patterns. In this framework, an objective measure called h-confidence is applied to discover hyperclique patterns. We prove that the items in a hyperclique pattern have a guaranteed level of global pairwise similarity to one another as measured by the cosine similarity (uncentered Pearson's correlation coefficient). Also, we show that the h-confidence measure satisfies a cross-support property which can help efficiently eliminate spurious patterns involving items with substantially different support levels. Indeed, this cross-support property is not limited to h-confidence and can be generalized to some other association measures. In addition, an algorithm called hyperclique miner is proposed to exploit both cross-support and anti-monotone properties of the h-confidence measure for the efficient discovery of hyperclique patterns. Finally, our experimental results show that hyperclique miner can efficiently identify hyperclique patterns, even at extremely low levels of support.

Keywords Association analysis · Hyperclique patterns · Pattern Mining

H. Xiong (✉)
Department of Management Science & Information Systems, Rutgers University, Ackerson Hall,
180 University Avenue, Newark, NJ 07102
e-mail: hui@rbs.rutgers.edu

P.-N. Tan
Department of Computer Science & Engineering, Michigan State University, 3115 Engineering
Building, East Lansing, MI 48824-1226
e-mail: ptan@cse.msu.edu

V. Kumar
Department of Computer Science and Engineering, University of Minnesota, 4-192, EE/CSci Building,
Minneapolis, MN 55455
e-mail: kumar@cs.umn.edu

1. Introduction

Many data sets have inherently skewed support distributions. For example, the frequency distribution of English words appearing in text documents is highly skewed—while a few of the words may appear many times, most of the words appear only a few times. Such a distribution has also been observed in other application domains, including retail data, Web click-streams, and telecommunication data.

This paper examines the problem of mining association patterns (Agrawal et al., 1993) from data sets with skewed support distributions. Most of the algorithms developed so far rely on the support-based pruning strategy to prune the combinatorial search space. However, this strategy is not effective for data sets with skewed support distributions due to the following reasons.

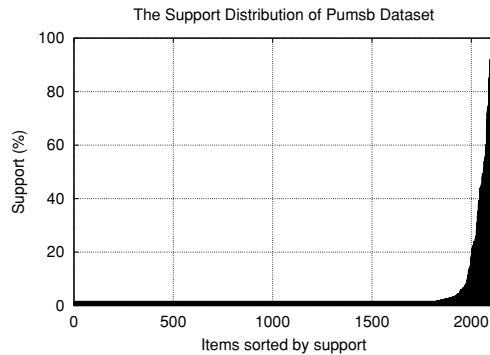
- If the minimum support threshold is low, we may extract too many *spurious patterns involving items with substantially different support levels*. We call such patterns as weakly-related *cross-support* patterns. For example, {Caviar, Milk} is a possible weakly-related cross-support pattern since the support for an expensive item such as Caviar is expected to be much lower than the support for an inexpensive item such as Milk. Such patterns are spurious because they tend to be poorly correlated. Using a low minimum support threshold also increases the computational and memory requirements of current state-of-the-art algorithms considerably.
- If the minimum support threshold is high, we may miss many *interesting patterns occurring at low levels of support* (Hastie et al., 2001). Examples of such patterns are associations among rare but expensive items such as caviar and vodka, gold necklaces and earrings, or TVs and DVD players.

As an illustration, consider the `pumsb` census data set,¹ which is often used as a benchmark data set for evaluating the computational performance of association rule mining algorithms. Figure 1 shows the skewed nature of the support distribution. Note that 81.5% of the items have support less than 0.01 while only 0.95% of them having support greater than 0.9.

Table 1 shows a partitioning of these items into five disjoint groups based on their support levels. The first group, *S1*, has the lowest support level (less than or equal to 0.01) but contains the most number of items (i.e., 1735 items). In order to detect patterns involving items from *S1*, we need to set the minimum support threshold to be less than 0.01. However, such a low support threshold will degrade the performance of existing algorithms considerably. For example, our experiments showed that, when applied to the `pumsb` data set at support threshold less than 0.4,² state-of-the-art algorithms such as Apriori (Agrawal and Srikant, 1994) and Charm (Zaki and Hsiao, 2002) break down due to excessive memory requirements. Even if a machine with unlimited memory is provided, such algorithms can still produce a large number of weakly-related cross-support patterns when the support threshold is low. Just to give an indication of the scale, out of the 18847 frequent pairs involving items from *S1* and *S5* at support level 0.0005, about 93% of them are cross-support patterns, i.e., containing items from both *S1* and *S5*. The pair wise correlations within these cross-support patterns are extremely poor because the presence of the item from *S5* does not necessarily imply the presence of the item from *S1*. Indeed, the maximum correlation obtained from these cross-support patterns is only 0.029377. In contrast, item pairs from *S1* alone or *S5* alone

¹ It is available at <http://www.almaden.ibm.com/software/quest/resources>.

² This is observed on Sun Ultra 10 work station with a 440 MHz CPU and 128 Mbytes of memory.

Fig. 1 The support distribution of Pumsb

have correlation as high as 1.0. The above discussion suggests that it will be advantageous to develop techniques that can automatically eliminate such patterns during the mining process.

Indeed, the motivation for this work is to strike a balance between the ability to detect patterns at very low support levels and the ability to remove spurious associations among items with substantially different support levels. A naive approach for doing this is to apply a very low minimum support threshold during the association mining step, followed by a post-processing step to eliminate spurious patterns. This approach may fail due to the following reasons: (i) the computation cost can be very high due to the large number of patterns that need to be generated; (ii) current algorithms may break down at very low support thresholds due to excessive memory requirements. Although there have been recent attempts to efficiently extract interesting patterns without using support thresholds, they do not guarantee the completeness of the discovered patterns. These methods include sampling (Cohen et al., 2000) and other approximation schemes (Yang et al., 2001).

A better approach will be to have a measure that can efficiently identify useful patterns even at low levels of support and can be used to automatically remove spurious patterns during the association mining process. Omiecinski recently introduced a measure called *all-confidence* (Omiecinski, 2003) as an alternative to the support measure. The all-confidence measure is computed by taking the minimum confidence of all association rules generated from a given itemset. Omiecinski proved that all-confidence has the desirable anti-monotone property and incorporated this property directly into the mining process for efficient computation of all patterns with sufficiently high value of all-confidence. Note that we had independently proposed a measure called *h-confidence* (Xiong et al., 2003a) and had named the itemsets discovered by the h-confidence measure as hyperclique patterns. As shown in Section 2, h-confidence and all-confidence measures are equivalent. To maintain consistent notation and presentation, we will use the term h-confidence in the remainder of this paper.

Table 1 Groups of items for pumsb data set

Group	S_1	S_2	S_3	S_4	S_5
Support	0–0.01	0.01–0.05	0.05–0.4	0.4–0.9	0.9–1.0
# Items	1735	206	101	51	20

1.1. Contributions

This paper extends our work (Xiong et al., 2003b) on hyperclique pattern discovery and makes the following contributions. First, we formally define the concept of the cross-support property, which helps efficiently eliminate spurious patterns involving items with substantially different support levels. We show that this property is not limited to h-confidence and can be generalized to some other association measures. Also, we provide an algebraic cost model to analyze the computation savings obtained by the cross-support property. Second, we prove that if a hyperclique pattern has an h-confidence value above the minimum h-confidence threshold, h_c , then every pair of objects within the hyperclique pattern must have a cosine similarity (uncentered Pearson's correlation coefficient³) greater than or equal to h_c . Also, we show that all derived size-2 hyperclique patterns are guaranteed to be positively correlated, as long as the minimum h-confidence threshold is above the maximum support of all items in the given data set. Finally, we refine our algorithm (called hyperclique miner) that is used to discover hyperclique patterns. Our experimental results show that hyperclique miner can efficiently identify hyperclique patterns, even at low support levels. In addition, we demonstrate that the utilization of the cross-support property provides significant additional pruning over that provided by the anti-monotone property of h-confidence.

1.2. Related work

Recently, there has been growing interest in developing techniques for mining association patterns without support constraints. For example, Wang et al. (2001) proposed the use of universal-existential upward closure property of confidence to extract association rules without specifying the support threshold. However, this approach does not explicitly eliminate cross-support patterns. Cohen et al. (2000) have proposed using the Jaccard similarity measure, $\text{sim}(x, y) = \frac{P(x \cap y)}{P(x \cup y)}$, to capture interesting patterns without using a minimum support threshold. As we show in Section 3, the Jaccard measure has the cross-support property which can be used to eliminate cross-support patterns. However, the discussion in Cohen et al. (2000) focused on how to employ a combination of random sampling and hashing techniques for efficiently finding highly-correlated pairs of items.

Many alternative techniques have also been developed to push various types of constraints into the mining algorithm (Bayardo et al., 1999; Grahne et al., 2000; Liu et al., 1999). Although these approaches may greatly reduce the number of patterns generated and improve computational performance by introducing additional constraints, they do not offer any specific mechanism to eliminate weakly-related patterns involving items with different support levels.

Besides all-confidence (Omicinski, 2003), other measures of association have been proposed to extract interesting patterns in large data sets. For example, Brin et al. (1997) introduced the *interest* measure and χ^2 test to discover patterns containing highly dependent items. However, these measures do not possess the desired anti-monotone property.

The concept of closed itemsets (Pei et al., 2000; Zaki and Hsiao, 2002) and maximal itemsets (Bayardo, 1998; Burdick et al., 2001) have been developed to provide a compact presentation of frequent patterns and of the "boundary" patterns, respectively. Algorithms for computing closed itemsets and maximal itemsets are often much more efficient than those for computing frequent patterns, especially for dense data sets, and thus may be

³ When computing Pearson's correlation coefficient, the data mean is not subtracted.

able to work with lower support thresholds. Hence, it may appear that one could discover closed or maximal itemsets at low levels of support, and then perform a post-processing step to eliminate cross-support patterns represented by these concepts. However, as shown by our experiments, for data sets with highly skewed support distributions, the number of spurious patterns represented by maximal or closed itemsets is still very large. This makes the computation cost of post-processing very high.

1.3. Overview

The remainder of this paper is organized as follows. Section 2 defines the concept of hyperclique patterns and shows the equivalence between all-confidence and h-confidence. In Section 3, we introduce the concept of the cross-support property and prove that the h-confidence measure has this property. We demonstrate the relationship between the h-confidence measure and some other association measures in Section 4. Section 5 describes the hyperclique miner algorithm. In Section 6, we present experimental results. Finally, Section 7 gives our conclusions and suggestions for future work.

2. Hyperclique pattern

In this section, we present a formal definition of hyperclique patterns and show the equivalence between all-confidence (Omiecinski, 2003) and h-confidence.

2.1. Hyperclique pattern definition

A hypergraph $H = \{V, E\}$ consists of a set of vertices V and a set of hyperedges E . The concept of a hypergraph extends the conventional definition of a graph in the sense that each hyperedge can connect more than two vertices. It also provides an elegant representation for association patterns, where every pattern (itemset) P can be modeled as a hypergraph with each item $i \in P$ represented as a vertex and a hyperedge connecting all the vertices of P . A hyperedge can also be weighted in terms of the magnitude of relationships among items in the corresponding itemset. In the following, we define a metric called h-confidence as a measure of association for an itemset.

Definition 1. The **h-confidence** of an itemset $P = \{i_1, i_2, \dots, i_m\}$ is defined as follows:

$$hconf(P) = \min\{conf\{i_1 \rightarrow i_2, \dots, i_m\}, conf\{i_2 \rightarrow i_1, i_3, \dots, i_m\}, \dots, conf\{i_m \rightarrow i_1, \dots, i_{m-1}\}\},$$

where *conf* follows from the conventional definition of association rule confidence (Agrawal et al., 1993).

Example 1. Consider an itemset $P = \{A, B, C\}$. Assume that $supp(\{A\}) = 0.1$, $supp(\{B\}) = 0.1$, $supp(\{C\}) = 0.06$, and $supp(\{A, B, C\}) = 0.06$, where *supp* denotes the

support (Agrawal et al., 1993) of an itemset. Since

$$\begin{aligned} \text{conf}\{A \rightarrow B, C\} &= \text{supp}(\{A, B, C\})/\text{supp}(\{A\}) = 0.6, \\ \text{conf}\{B \rightarrow A, C\} &= \text{supp}(\{A, B, C\})/\text{supp}(\{B\}) = 0.6, \\ \text{conf}\{C \rightarrow A, B\} &= \text{supp}(\{A, B, C\})/\text{supp}(\{C\}) = 1, \end{aligned}$$

therefore, $h\text{conf}(P) = \min\{0.6, 0.6, 1\} = 0.6$.

Definition 2. Given a set of items $I = \{i_1, i_2, \dots, i_n\}$ and a minimum h-confidence threshold h_c , an itemset $P \subseteq I$ is a **hyperclique pattern** if and only if $h\text{conf}(P) \geq h_c$.

A hyperclique pattern P can be interpreted as follows: the presence of any item $i \in P$ in a transaction implies the presence of all other items $P - \{i\}$ in the same transaction with probability at least h_c . This suggests that h-confidence is useful for capturing patterns containing items which are strongly related with each other, especially when the h-confidence threshold is sufficiently large.

Nevertheless, the hyperclique pattern mining framework may miss some interesting patterns too. For example, an itemset such as $\{A, B, C\}$ may have very low h-confidence, and yet it may be still interesting if one of its rules, say $AB \rightarrow C$, has very high confidence. Discovering such type of patterns is beyond the scope of this paper.

2.2. The equivalence between the all-confidence measure and the h-confidence measure

The following is a formal definition of the all-confidence measure as given in Omiecinski (2003).

Definition 3. The all-confidence measure (Omiecinski, 2003) for an itemset $P = \{i_1, i_2, \dots, i_m\}$ is defined as $\text{allconf}(P) = \min\{\{\text{conf}(A \rightarrow B \mid \forall A, B \subset P, A \cup B = P, A \cap B = \emptyset)\}$.

Conceptually, the all-confidence measure checks every association rule extracted from a given itemset. This is slightly different from the h-confidence measure, which examines only rules of the form $\{i\} \rightarrow P - \{i\}$, where there is only one item on the left-hand side of the rule. Despite their syntactic difference, both measures are mathematically identical to each other, as shown in the lemma below.

Lemma 1. For an itemset $P = \{i_1, i_2, \dots, i_m\}$, $h\text{conf}(P) \equiv \text{allconf}(P)$.

Proof: The confidence for any association rule $A \rightarrow B$ extracted from an itemset P is given by $\text{conf}\{A \rightarrow B\} = \frac{\text{supp}(A \cup B)}{\text{supp}(A)} = \frac{\text{supp}(P)}{\text{supp}(A)}$. From Definition 3, we may write $\text{allconf}(P) = \min(\{\text{conf}\{A \rightarrow B\}\}) = \frac{\text{supp}(P)}{\max(\{\text{supp}(A) \mid \forall A \subset P\})}$. From the anti-monotone property of the support measure, $\max(\{\text{supp}(A) \mid A \subset P\}) = \max_{1 \leq k \leq m} \{\text{supp}(\{i_k\})\}$. Hence,

$$\text{allconf}(P) = \frac{\text{supp}(\{i_1, i_2, \dots, i_m\})}{\max_{1 \leq k \leq m} \{\text{supp}(\{i_k\})\}}. \quad (1)$$

Also, we simplify h-confidence for an itemset P in the following way.

$$\begin{aligned}
 hconf(P) &= \min\{conf\{i_1 \rightarrow i_2, \dots, i_m\}, conf\{i_2 \rightarrow i_1, i_3, \dots, i_m\}, \dots, conf\{i_m \rightarrow i_1, \dots, i_{m-1}\}\} \\
 &= \min \left\{ \frac{supp(\{i_1, i_2, \dots, i_m\})}{supp(\{i_1\})}, \frac{supp(\{i_1, i_2, \dots, i_m\})}{supp(\{i_2\})}, \dots, \frac{supp(\{i_1, i_2, \dots, i_m\})}{supp(\{i_m\})} \right\} \\
 &= supp(\{i_1, \dots, i_m\}) \cdot \min \left\{ \frac{1}{supp(\{i_1\})}, \frac{1}{supp(\{i_2\})}, \dots, \frac{1}{supp(\{i_m\})} \right\} \\
 &= \frac{supp(\{i_1, i_2, \dots, i_m\})}{\max_{1 \leq k \leq m} \{supp(\{i_k\})\}}.
 \end{aligned}$$

The above expression is identical to Eq. (1), so Lemma 1 holds. □

2.3. Anti-monotone property of h-confidence

Omicinski has previously shown that the all-confidence measure has the anti-monotone property (Omicinski, 2003). In other words, if the all-confidence of an itemset P is greater than a user-specified threshold, so is every subset of P . Since h-confidence is mathematically identical to all-confidence, it is also monotonically non-increasing as the size of the hyperclique pattern increases. Such a property allows us to push the h-confidence constraint directly into the mining algorithm. Specifically, when searching for hyperclique patterns, the algorithm eliminates every candidate pattern of size m having at least one subset of size $m - 1$ that is not a hyperclique pattern.

Besides being anti-monotone, the h-confidence measure also possesses other desirable properties. A detailed examination of these properties is presented in the following sections.

3. The cross-support property

In this section, we describe the cross-support property of h-confidence and explain how this property can be used to efficiently eliminate cross-support patterns. Also, we show that the cross-support property is not limited to h-confidence and can be generalized to some other association measures. Finally, a sufficient condition is provided for verifying whether a measure satisfies the cross-support property or not.

3.1. Illustration of the cross-support property

First, a formal definition of cross-support patterns is given as follows.

Definition 4 (Cross-support Patterns). Given a threshold t , a pattern P is a cross-support pattern with respect to t if P contains two items x and y such that $\frac{supp(\{x\})}{supp(\{y\})} < t$, where $0 < t < 1$.

Let us consider the diagram shown in Fig. 2, which illustrates cross-support patterns in a hypothetical data set. In the figure, the horizontal axis shows items sorted by support in non-decreasing order and the vertical axis shows the corresponding support for items. For example, in the figure, the pattern $\{x, y, j\}$ is a cross-support pattern with respect to the threshold $t = 0.6$, since this pattern contains two items x and y such that $\frac{supp(\{x\})}{supp(\{y\})} = \frac{0.3}{0.6} = 0.5 < t = 0.6$.

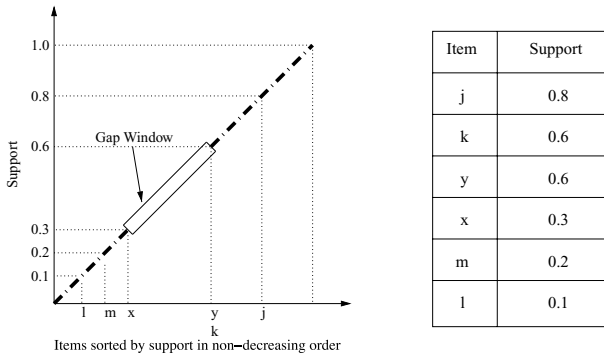


Fig. 2 An example to illustrate the cross-support pruning

Once we have the understanding of cross-support patterns, we present the cross-support property of h-confidence in the following lemma.

Lemma 2 (Cross-support Property of the h-confidence measure). Any cross-support pattern P with respect to a threshold t is guaranteed to have $hconf(P) < t$.

Proof: Since P is a cross-support pattern with respect to the threshold t , by Definition 4, we know P contains at least two items x and y such that $\frac{supp(\{x\})}{supp(\{y\})} < t$, where $0 < t < 1$. Without loss of generality, let $P = \{\dots, x, \dots, y, \dots\}$. By Eq. (1), we have the following.

$$\begin{aligned}
 hconf(P) &= \frac{supp(P)}{\max\{\dots, supp(\{x\}), \dots, supp(\{y\}), \dots\}} \\
 &\leq \frac{supp(\{x\})}{\max\{\dots, supp(\{x\}), \dots, supp(\{y\}), \dots\}} \leq \frac{supp(\{x\})}{supp(\{y\})} < t
 \end{aligned}$$

Note that the anti-monotone property of support is applied to the numerator part of the h-confidence expression in the above proof. □

Corollary 1. Given an item y , all patterns that contain y and at least one item with support less than $t \cdot supp(\{y\})$ (for $0 < t < 1$) are cross-support patterns with respect to t and are guaranteed to have h-confidence less than t . All such patterns can be automatically eliminated without computing their h-confidence if we are only interested in patterns with h-confidence greater than t .

Proof: This corollary follows from Definition 4 and Lemma 2. □

For a given h-confidence threshold, the above Corollary provides a systematic way to find and eliminate candidate itemsets that are guaranteed not to be hyperclique patterns. In the following, we present an example to illustrate the cross-support pruning.

Example 2. Figure 2 shows the support distributions of five items and their support values. By Corollary 1, given a minimum h-confidence threshold $h_c = 0.6$, all patterns contain item y and at least one item with support less than $supp(\{y\}) \cdot h_c = 0.6 \times 0.6 = 0.36$ are cross-support patterns and are guaranteed to have h-confidence less than 0.6. Hence, all patterns

containing item y and at least one of l, m, x do not need to be considered if we are only interested in patterns with h-confidence greater than 0.6.

3.2. Generalization of the cross-support property

In this subsection, we generalize the cross-support property to other association measures. First, we give a generalized definition of the cross-support property.

Definition 5 (Generalized Cross-support Property). Given a measure f , for any cross-support pattern P with respect to a threshold t , if there exists a monotone increasing function g such that $f(P) < g(t)$, then the measure f has the cross-support property.

Given the h-confidence measure, for any cross-support pattern P with respect to a threshold t , if we let the function $g(l) = l$, then we have $hconf(P) < g(t) = t$ by Lemma 2. Hence, the h-confidence measure has the cross-support property. Also, if a measure f has the cross-support property, the following theorem provides a way to automatically eliminate cross-support patterns.

Theorem 1. *Given an item y , a measure f with the cross-support property, and a threshold θ , any pattern P that contains y and at least one item x with support less than $g^{-1}(\theta) \cdot supp(\{y\})$ is a cross-support pattern with respect to the threshold $g^{-1}(\theta)$ and is guaranteed to have $f(P) < \theta$, where g is the function to make the measure f satisfy the generalized cross-support property.*

Proof: Since $supp(\{x\}) < g^{-1}(\theta) \cdot supp(\{y\})$, we have $\frac{supp(\{x\})}{supp(\{y\})} < g^{-1}(\theta)$. By Definition 4, the pattern P is a cross-support pattern with respect to the threshold $g^{-1}(\theta)$. Also, because the measure f has the cross-support property, by Definition 5, there exists a monotone increasing function g such that $f(P) < g(g^{-1}(\theta)) = \theta$. Hence, for the given threshold θ , all such patterns can be automatically eliminated if we are only interested in patterns with the measure f greater than θ . □

Table 2 shows two measures that have the *cross-support* property. The corresponding monotone increasing functions for these measures are also shown in this table. However, some measures, such as support and odds ratio (Tan et al. 2002), do not possess such a property.

Table 2 Examples of measures of association that have the *cross-support* property (assuming that $supp(\{x\}) < supp(\{y\})$)

Measure	Computation formula	Upper bound	Function
Cosine	$\frac{supp(\{x, y\})}{\sqrt{supp(\{x\})supp(\{y\})}}$	$\sqrt{\frac{supp(\{x\})}{supp(\{y\})}}$	$g(l) = \sqrt{l}$
Jaccard	$\frac{supp(\{x, y\})}{supp(\{x\}) + supp(\{y\}) - supp(\{x, y\})}$	$\frac{supp(\{x\})}{supp(\{y\})}$	$g(l) = l$

3.3. Discussion

In this subsection, we presents some analytical results for the amount of computational savings obtained by the cross-support property of h-confidence. To facilitate our discussion, we only analyze the amount of computational savings for size-2 hyperclique patterns. We first introduce the definitions of several concepts.

Definition 6. The pruning ratio is defined as follows

$$\gamma(\theta) = \frac{S(\theta)}{T}, \quad (2)$$

where θ is the minimum h-confidence threshold, $S(\theta)$ is the number of item pairs which are pruned by the cross-support property at the minimum h-confidence threshold θ , and T is the total number of item pairs in the database. For a given database, T is a fixed number and is equal to $\binom{n}{2} = \frac{n(n-1)}{2}$, where n is the number of items.

Definition 7. For a sorted item list, the rank-support function $f(k)$ is a function which presents the support in terms of the rank k .

For a given database, let $I = \{A_1, A_2, \dots, A_n\}$ be an item list sorted by item supports in non-increasing order. Then item A_1 has the maximum support and the rank-support function $f(k) = \text{supp}(A_k), \forall 1 \leq k \leq n$, which is monotone decreasing with the increase of the rank k . To quantify the computation savings for a given item A_j ($1 \leq j < n$) at the threshold θ , we need to find only the first item A_l ($j < l \leq n$) such that $\text{supp}(A_l)/\text{supp}(A_j) < \theta$. Then, we know that $\text{supp}(A_i)/\text{supp}(A_j) < \theta$, where $l \leq i \leq n$. By Corollary 1, all these $n - l + 1$ pairs can be pruned by the cross-support property since,

$$\text{supp}(A_l)/\text{supp}(A_j) = f(l)/f(j) < \theta$$

Also, if the rank-support function $f(k)$ is monotone decreasing with the increase of the rank k , we get

$$l > f^{-1}(\theta f(j))$$

To make the computation simple, we let $l = f^{-1}(\theta f(j)) + 1$. Therefore, for a given item A_j ($1 < j \leq n$), the computation cost for $(n - f^{-1}(\theta f(j)))$ item pairs can be saved. As a result, the total computation savings is shown below.

$$S(\theta) = \sum_{j=2}^n \{n - f^{-1}(\theta f(j))\} \quad (3)$$

Finally, we conduct computation savings analysis on the case that data sets have a generalized Zipf distribution (Zipf, 1949). In this case, the rank-support function has a generalized Zipf distribution and $f(k) = \frac{c}{k^p}$, where c and p are constants and $p \geq 1$. When p is equal to 1, the rank-support function has a Zipf distribution.

Lemma 3. *When a database has a generalized Zipf rank-support distribution $f(k)$ and $f(k) = \frac{c}{k^p}$, for a user-specified minimum h-confidence threshold θ , the pruning ratio increases with the increase of p and the h-confidence threshold θ , where $0 < \theta \leq 1$.*

Proof: Since the rank-support function $f(k) = \frac{c}{k^p}$, the inverse function $f^{-1}(y) = (\frac{c}{y})^{\frac{1}{p}}$. Accordingly,

$$f^{-1}(\theta f(j)) = \left(\frac{c}{\theta \frac{c}{j^p}}\right)^{\frac{1}{p}} = \frac{j}{(\theta)^{\frac{1}{p}}}$$

Applying Eq. (3), we get:

$$\begin{aligned} S(\theta) &= \sum_{j=2}^n \{n - f^{-1}(\theta f(j))\} \\ &= n(n - 1) - \sum_{j=2}^n \frac{j}{(\theta)^{\frac{1}{p}}} \\ &= n(n - 1) - \frac{(n - 1)(n + 2)}{2} \frac{1}{\theta^{\frac{1}{p}}} \end{aligned}$$

Since the pruning ratio $\gamma(\theta) = \frac{S(\theta)}{T}$ and $T = \frac{n(n-1)}{2}$,

$$\Rightarrow \gamma(\theta) = 2 - \frac{n + 2}{n} \frac{1}{\theta^{\frac{1}{p}}}$$

Thus, we can derive two rules as follows:

$$\begin{aligned} \text{rule 1 : } \theta \nearrow &\Rightarrow \frac{n + 2}{n} \frac{1}{\theta^{\frac{1}{p}}} \searrow \Rightarrow \gamma(\theta) \nearrow \\ \text{rule 2 : } p \nearrow &\Rightarrow \frac{n + 2}{n} \frac{1}{\theta^{\frac{1}{p}}} \searrow \Rightarrow \gamma(\theta) \nearrow \end{aligned}$$

□

Therefore, the claim that the pruning ratio increases with the increase of p and the h-confidence threshold θ holds. Also, with the increase of p , the generalized Zipf distribution becomes more skewed. In other words, the pruning effect of the cross-support property become more significant for data sets with more skewed support distributions.

4. The h-confidence as a measure of association

In this section, we first show that the strength or magnitude of a relationship described by h-confidence is consistent with the strength or magnitude of a relationship described by two

traditional association measures: the Jaccard measure (Rijsbergen, 1979) and the correlation coefficient (Reynolds, 1977).

Given a pair of items $P = \{i_1, i_2\}$, the affinity between both items using P these measures is defined as follows:

$$jaccard(P) = \frac{supp(\{i_1, i_2\})}{supp(\{i_1\}) + supp(\{i_2\}) - supp(\{i_1, i_2\})},$$

$$correlation, \phi(P) = \frac{supp(\{i_1, i_2\}) - supp(\{i_1\})supp(\{i_2\})}{\sqrt{supp(\{i_1\})supp(\{i_2\})(1 - supp(\{i_1\}))(1 - supp(\{i_2\}))}}.$$

Also, we demonstrate that h-confidence is a measure of association that can be used to capture the strength or magnitude of a relationship among several objects.

4.1. Relationship between h-confidence and Jaccard

In the following, for a size-2 hyperclique pattern P , we provide a lemma that gives a lower bound for $jaccard(P)$.

Lemma 4. *If an item set $P = \{i_1, i_2\}$ is a size-2 hyperclique pattern, then $jaccard(P) \geq h_c/2$.*

Proof: By Eq. (1), $hconf(P) = \frac{supp(\{i_1, i_2\})}{\max\{supp(\{i_1\}), supp(\{i_2\})\}}$. Without loss of generality, let $supp(\{i_1\}) \geq supp(\{i_2\})$. Given that P is a hyperclique pattern, $hconf(P) = \frac{supp(\{i_1, i_2\})}{supp(\{i_1\})} \geq h_c$. Furthermore, since $supp(\{i_1\}) \geq supp(\{i_2\})$, $jaccard(P) = \frac{supp(\{i_1, i_2\})}{supp(\{i_1\}) + supp(\{i_2\}) - supp(\{i_1, i_2\})} \geq \frac{supp(\{i_1, i_2\})}{2supp(\{i_1\})} \geq h_c/2$. \square

Lemma 4 suggests that if the h-confidence threshold h_c is sufficiently high, then all size-2 hyperclique patterns contain items that are strongly related with each other in terms of the Jaccard measure, since the Jaccard values of these hyperclique patterns are bounded from below by $h_c/2$.

4.2. Relationship between h-confidence and correlation

In this subsection, we illustrate the relationship between h-confidence and Pearson's correlation. More specifically, we show that if at least one item in a size-2 hyperclique pattern has a support value less than the minimum h-confidence threshold, h_c , then two items within this hyperclique pattern must be positively correlated.

Lemma 5. *Let S be a set of items and h_c be the minimum h-confidence threshold, we can form two item groups: S_1 and S_2 such that $S_1 = \{x | supp(\{x\}) < h_c \text{ and } x \in S\}$ and $S_2 = \{y | supp(\{y\}) \geq h_c \text{ and } y \in S\}$. Then, any size-2 hyperclique pattern $P = \{A, B\}$ has a positive correlation coefficient in each of the following cases: Case 1: $A \in S_1$ and $B \in S_2$. Case 2: $A \in S_1$ and $B \in S_1$.*

Proof: For a size-2 hyperclique pattern $P = \{A, B\}$, without loss of generality, we assume that $supp(\{A\}) \leq supp(\{B\})$. Since $hconf(P) \geq h_c$, we know $\frac{supp(\{A, B\})}{\max\{supp(\{A\}), supp(\{B\})\}} \geq h_c$. In other words, $supp(\{A, B\}) \geq h_c \cdot supp(\{B\})$.

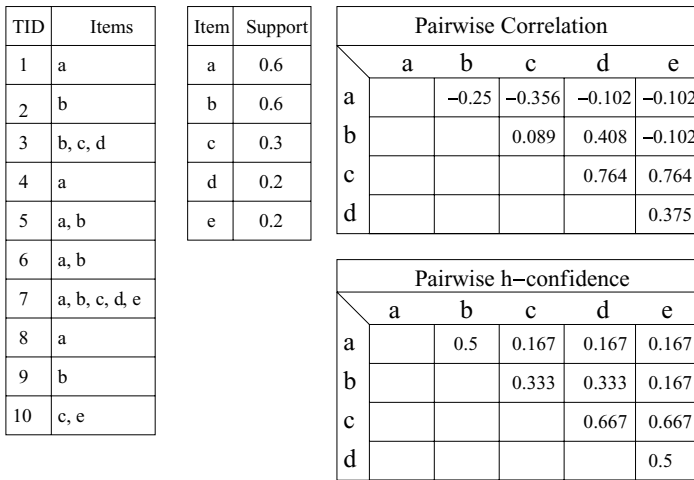


Fig. 3 Illustration of the relationship between h-confidence and correlation

From the definition of Pearson’s correlation coefficient:

$$\phi(\{A, B\}) = \frac{supp(\{A, B\}) - supp(\{A\})supp(\{B\})}{\sqrt{supp(\{A\})supp(\{B\})(1 - supp(\{A\}))(1 - supp(\{B\}))}}$$

Also, $\sqrt{supp(\{A\})(1 - supp(\{A\}))} \leq (supp(\{A\}) + 1 - supp(\{A\}))/2 = 1/2$. so, $\phi(\{A, B\}) \geq 4(supp(\{A, B\}) - supp(\{A\})supp(\{B\})) \geq 4supp(\{B\})(h_c - supp(\{A\}))$

Case 1: if $A \in S_1$ and $B \in S_2$

Since $A \in S_1$ and $B \in S_2$, we know $supp(\{A\}) < h_c$ due to the way that we construct S_1 and S_2 . As a result, $4supp(\{B\})(h_c - supp(\{A\})) > 0$. Hence, $\phi > 0$.

Case 2: if $A \in S_1$ and $B \in S_1$

Since $A \in S_1$, we know $supp(\{A\}) < h_c$ due to the way that we construct S_1 and S_2 . As a result, $4supp(\{B\})(h_c - supp(\{A\})) > 0$. Hence, $\phi > 0$. □

Example 3. Figure 3 illustrates the relationship between h-confidence and correlation. Assume that the minimum h-confidence threshold is 0.45. In the figure, there are four pairs including $\{a, b\}$, $\{c, d\}$, $\{c, e\}$, $\{d, e\}$ with h-confidence greater than 0.45. Among these four pairs, $\{c, d\}$, $\{c, e\}$, $\{d, e\}$ contain at least one item with the support value less than the h-confidence threshold, 0.45. By Lemma 5, all these three pairs have positive correlation. Furthermore, if we increase the minimum h-confidence threshold to be greater than 0.6 that is the maximum support of all items in the given data set, all size-2 hyperclique patterns are guaranteed to have positive correlation.

In practice, many real-world data sets, such as the point-of-sale data collected at department stores, contain very few items with considerably high support. For instance, the retail data set used in our own experiment contains items with a maximum support equals to 0.024. Following the discussion presented above, if we set the minimum h-confidence threshold above 0.024, all size-2 hyperclique patterns are guaranteed to be positively correlated. To make this discussion more interesting, recall the well-known coffee-tea example given in

(Brin et al., 1997). This example illustrates the drawback of using confidence as a measure of association. Even though the confidence for the rule $tea \rightarrow coffee$ may be high, both items are in fact negatively-correlated with each other. Hence, the confidence measure can be misleading. Instead, with h-confidence, we may ensure that all derived size-2 patterns are positively correlated, as long as the minimum h-confidence threshold is above the maximum support of all items.

4.3. H-confidence for measuring the relationship among several objects

In this subsection, we demonstrate that the h-confidence measure can be used to describe the strength or magnitude of a relationship among several objects.

Given a pair of items $P = \{i_1, i_2\}$, the cosine similarity between both items is defined as follows.

$$\text{cosine}(P) = \frac{\text{supp}(\{i_1, i_2\})}{\sqrt{\text{supp}(\{i_1\})\text{supp}(\{i_2\})}},$$

Note that the cosine similarity is also known as uncentered Pearson's correlation coefficient (when computing Pearson's correlation coefficient, the data mean is not subtracted). For a size-2 hyperclique pattern P , we first derive a lower bound for the cosine similarity of the pattern P , $\text{cosine}(P)$, in terms of the minimum h-confidence threshold h_c .

Lemma 6. *If an item set $P = \{i_1, i_2\}$ is a size-2 hyperclique pattern, then $\text{cosine}(P) \geq h_c$.*

Proof: By Eq. (1), $\text{hconf}(P) = \frac{\text{supp}(\{i_1, i_2\})}{\max\{\text{supp}(\{i_1\}), \text{supp}(\{i_2\})\}}$. Without loss of generality, let $\text{supp}(\{i_1\}) \geq \text{supp}(\{i_2\})$. Given that P is a hyperclique pattern, $\text{hconf}(P) = \frac{\text{supp}(\{i_1, i_2\})}{\text{supp}(\{i_1\})} \geq h_c$. Since $\text{supp}(\{i_1\}) \geq \text{supp}(\{i_2\})$, $\text{cosine}(P) = \frac{\text{supp}(\{i_1, i_2\})}{\sqrt{\text{supp}(\{i_1\})\text{supp}(\{i_2\})}} \geq \frac{\text{supp}(\{i_1, i_2\})}{\text{supp}(\{i_1\})} \geq h_c$. \square

Lemma 6 suggests that if the h-confidence threshold h_c is sufficiently high, then all size-2 hyperclique patterns contain items that are strongly related with each other in terms of the cosine measure, since the cosine values of these hyperclique patterns are bounded from below by h_c .

For the case that hyperclique patterns have more than two objects, the following theorem guarantees that if a hyperclique pattern has an h-confidence value above the minimum h-confidence threshold, h_c , then every pair of objects within the hyperclique pattern must have a cosine similarity great than or equal to h_c .

Theorem 2. *Given a hyperclique pattern $P = \{i_1, i_2, \dots, i_k\}$ ($k > 2$) at the h-confidence threshold h_c , for any size-2 itemset $Q = \{i_l, i_m\}$ such that $Q \subset P$, we have $\text{cosine}(Q) \geq h_c$.*

Proof: By the anti-monotone property of the h-confidence measure and the condition that $Q \subset P$, we know Q is also a hyperclique pattern. Then, by Lemma 6, we know $\text{cosine}(Q) \geq h_c$. \square

Clique View. Indeed, a hyperclique pattern can be viewed as a clique, if we construct a graph in the following way. Treat each object in a hyperclique pattern as a vertex and put an edge between two vertices if the cosine similarity between two objects is above the h-confidence threshold, h_c . According to Theorem 2, there will be an edge between any two objects within a hyperclique pattern. As a result, a hyperclique pattern is a clique.

Viewed as cliques, hyperclique patterns have applications in many different domains. For instance, Xiong et al. (2004) show that the hyperclique pattern is the best candidate for pattern preserving clustering—a new paradigm for pattern based clustering. Also, Xiong et al. (2005) describe the use of hyperclique pattern discovery for identifying functional modules in protein complexes.

5. Hyperclique miner algorithm

The Hyperclique Miner is an algorithm which can generate all hyperclique patterns with support and h-confidence above user-specified minimum support and h-confidence thresholds.

Hyperclique Miner

Input:

- (1) a set F of K Boolean feature types $F = \{f_1, f_2, \dots, f_K\}$
- (2) a set T of N transactions $T = \{t_1 \dots t_N\}$, each $t_i \in T$ is a record with K attributes $\{i_1, i_2, \dots, i_K\}$ taking values in $\{0, 1\}$, where the i_p ($1 \leq p \leq K$) is the Boolean value for the feature type f_p .
- (3) A user specified minimum h-confidence threshold (h_c)
- (4) A user specified minimum support threshold (min_supp)

Output:

hyperclique patterns with h-confidence $> h_c$ and support $> min_supp$

Method:

- (1) Get size-1 prevalent items
- (2) **for** the size of itemsets in $(2, 3, \dots, K - 1)$ **do**
- (3) Generate candidate hyperclique patterns using the *generalized apriori_gen* algorithm
- (4) Generate hyperclique patterns
- (5) **end**;

Explanation of the detailed steps of the algorithm

Step 1 scans the database and gets the support for every item. Items with support above min_supp form size-1 candidate set C_1 . The h-confidence values for size-1 itemsets are 1.

All items in the set C_1 are sorted by the support values and relabeled in alphabetic order.

Step 2 to *Step 4* loops through 2 to $K - 1$ to generate qualified hyperclique patterns of size 2 or more. It stops whenever an empty candidate set of some size is generated.

Step 3 uses *generalized apriori_gen* to generate candidate hyperclique patterns of size k from hyperclique patterns of size $k - 1$. The *generalized apriori_gen* function is an adaptation of the *apriori_gen* function of the *Apriori* algorithm (Agrawal and Srikant, 1994). Let C_{k-1} indicate the set of all hyperclique patterns of size $k - 1$. The function works as follows. First, in the join step, we join C_{k-1} with C_{k-1} and get candidate set C_k . Next in the prune step, we delete all candidate hyperclique patterns $c \in C_k$ based on two major pruning techniques:

- (a) **Pruning based on anti-monotone property of h-confidence and support:** If any one of the $k - 1$ subsets of c does not belong to C_{k-1} , then c is pruned. (Recall that this prune step is also done in *apriori_gen* by Agrawal and Srikant because

of the anti-monotone property of support. Omiecinski (2003) also applied the anti-monotone property of all-confidence in his algorithm.)

- (b) **Pruning of cross-support patterns by using the cross-support property of h-confidence:** By Corollary 1, for the given h-confidence threshold h_c and an item y , all patterns that contain y and at least one item with support less than $h_c \cdot \text{supp}(\{y\})$ are cross-support patterns and are guaranteed to have h-confidence less than t . Hence, all such patterns can be automatically eliminated.

Note that the major pruning techniques applied in this step are illustrated by Example 4.

Step 4 computes exact support and h-confidence for all candidate patterns in C_k and prunes this candidate set using the user specified support threshold min_supp and the h-confidence threshold h_c . All remaining patterns are returned as hyperclique patterns of size k .

Example 4. Figure 4 illustrates the process of pruning the candidate generation step (Step 3) of the hyperclique miner algorithm. In this example, we assume the minimum support threshold to be zero and the minimum h-confidence threshold to be 0.6. Consider the state after all size-1 hyperclique patterns have been generated. Note that all these singleton items have support greater than 0. Also, by Eq. (1), h-confidence of all size-1 itemsets is 1.0, which

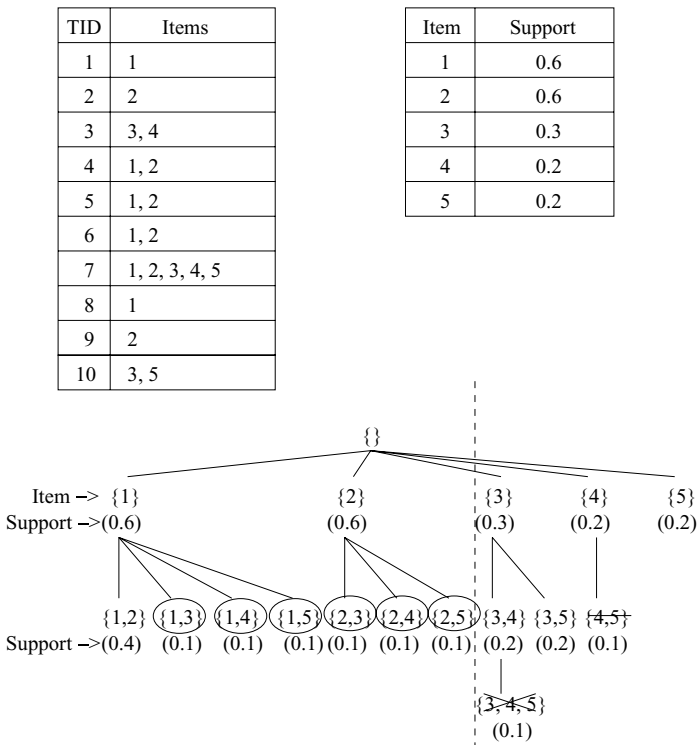


Fig. 4 A running example with a support threshold = 0 and an h-confidence threshold = 0.6. Note that crossed nodes are pruned by the anti-monotone property and circled nodes are pruned by the cross-support property

is greater than the user-specified h-confidence threshold of 0.6. Hence, all these singleton itemsets are hyperclique patterns.

There are two major pruning techniques that we can enforce in Step 3 while generating size-2 candidate itemsets from size-1 itemsets.

- (a) Pruning based on the anti-monotone property: No pruning is possible using this property since all size-1 itemsets are hyperclique patterns.
- (b) Pruning based on cross-support patterns by using the cross-support property: Given an h-confidence threshold 0.6, for the item 2, we can find an item 3 with $\text{supp}(\{3\}) = 0.3 < \text{supp}(\{2\}) \cdot 0.6 = 0.36$ in the sorted item list $\{1, 2, 3, 4, 5\}$. If we split this item list into two itemsets $L = \{1, 2\}$ and $U = \{3, 4, 5\}$, any pattern involving items from both L and U is a cross-support pattern with respect to the threshold 0.6. By Lemma 2, the h-confidence values for these cross-support patterns are less than 0.6. Since the h-confidence threshold is equal to 0.6, all cross-support patterns are pruned. In contrast, without applying cross-support pruning, we have to generate six cross-support patterns including $\{1, 3\}$, $\{1, 4\}$, $\{1, 5\}$, $\{2, 3\}$, $\{2, 4\}$, and $\{2, 5\}$ as candidate patterns and prune them later by computing the exact h-confidence values.

For the remaining size-2 candidate itemsets, the h-confidence of the itemset $\{4, 5\}$ is $\text{supp}(\{4, 5\})/\max\{\text{supp}(\{4\}), \text{supp}(\{5\})\} = 0.1/0.2 = 0.5$, which is less than the h-confidence threshold, 0.6. Hence, the itemset $\{4, 5\}$ is not a hyperclique pattern and is pruned.

Next, we consider pruning in Step 3 while generating size-3 candidate itemsets from size-2 hyperclique patterns.

- (c) Pruning based on the anti-monotone property. From the above, we know that the itemset $\{4, 5\}$ is not a hyperclique pattern. Then, we can prune the candidate pattern $\{3, 4, 5\}$ by the anti-monotone property of the h-confidence measure, since this pattern has one subset $\{4, 5\}$, which is not a hyperclique pattern.

6. Experimental evaluation

In this section, we present experiments to evaluate the performance of hyperclique miner and the quality of hyperclique patterns.

6.1. The experimental setup

6.1.1. Experimental data sets

Our experiments were performed on both real and synthetic data sets. Synthetic data sets were generated by using the IBM Quest synthetic data generator (Agrawal and Srikant, 1994), which gives us the flexibility of controlling the size and dimensionality of the database. A summary of the parameter settings used to create the synthetic data sets is presented in Table 3, where $|T|$ is the average size of a transaction, N is the number of items, and $|L|$ is the maximum number of potential frequent itemsets. Each data set contains 100000 transactions, with an average frequent pattern length equal to 4.

The real data sets are obtained from several application domains. Some characteristics of these data sets⁴ are shown in Table 4. In the table, the pumsb and pumsb* data sets

⁴ Note that the number of items shown in Table 4 for pumsb, pumsb* are somewhat different from the numbers reported in Zaki and Hsiao (2002), because we only consider item IDs for which the count is at least

Table 3 Parameter settings for synthetic data sets

Data set name	$ T $	$ L $	N	Size (MBytes)
T5.L100.N1000	5	100	1000	0.94
T5.L500.N5000	5	500	5000	2.48
T10.L1000.N10000	10	1000	10000	4.96
T20.L2000.N20000	20	2000	20000	10.73
T30.L3000.N30000	30	3000	30000	16.43
T40.L4000.N40000	40	4000	40000	22.13

Table 4 Real data set characteristics

Data set	#Item	#Record	Avg. length	Source
Pumsb	2113	49046	74	IBM Almaden
Pumsb*	2089	49046	50	IBM Almaden
LA1	29704	3204	145	TREC-5
Retail	14462	57671	8	Retail Store

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 collection⁵ and contains news articles from the Los Angeles Times. Finally, *retail* is a market-basket data set obtained from a large mail-order company.

6.1.2. Experimental platform

Our experiments were performed on a Sun Ultra 10 workstation with a 440 MHz CPU and 128 Mbytes of memory running the SunOS 5.7 operating system. We implemented hyperclique miner by modifying the publicly available Apriori implementation by Borgelt (<http://fuzzy.cs.uni-magdeburg.de/~borgelt>). When the h-confidence threshold is set to zero, the computational performance of hyperclique miner is approximately the same as the Borgelt's implementation of Apriori (Agrawal and Srikant, 1994).

6.2. The pruning effect of hyperclique miner

The purpose of this experiment is to demonstrate the effectiveness of the h-confidence pruning on hyperclique pattern generation. Note that hyperclique patterns can also be derived by first computing frequent patterns at very low levels of support, and using a post-processing step to eliminate weakly-related cross-support patterns. Hence, we use the conventional frequent pattern mining algorithms as the baseline to show the relative performance of hyperclique miner.

First, we evaluate the performance of hyperclique miner on the *LA1* data set. Figure 5(a) shows the number of patterns generated from the *LA1* data set at different h-confidence thresholds. As can be seen, at any fixed support threshold, the number of generated patterns increases quite dramatically with the decrease of the h-confidence threshold. For example,

one. For example, although the minimum item ID in *pumsb* is 0 and the maximum item ID is 7116, there are only 2113 distinct item IDs that appear in the data set.

⁵ The data set is available at <http://trec.nist.gov>.

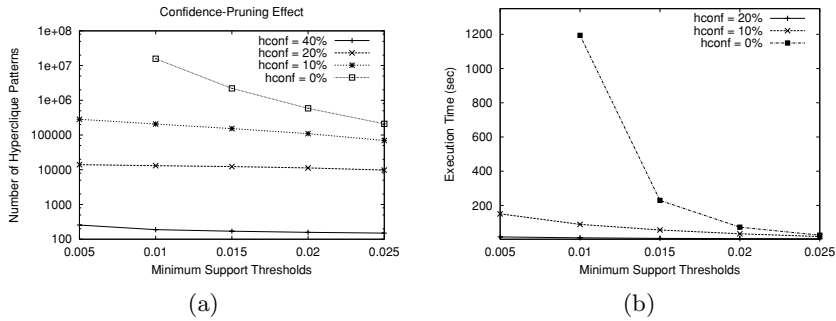


Fig. 5 (a) Number of patterns generated by hyperclique miner on LA1 data set. (b) The execution time of hyperclique miner on LA1 data set

when the support threshold is 0.01 and the h-confidence threshold is zero, the number of patterns generated is greater than 10^7 . In contrast, there are only several hundred hyperclique patterns when the h-confidence threshold is increased to 40%. Recall that, when the h-confidence threshold is equal to zero, the hyperclique miner essentially becomes the Apriori algorithm, as it finds all frequent patterns above certain support thresholds. As shown in Fig. 5(a), Apriori is not able to find frequent patterns at support level less than 0.01 due to excessive memory requirements. This is caused by the rapid increase of the number of patterns generated as the support threshold is decreased. On the other hand, for an h-confidence threshold greater than or equal to 10%, the number of patterns generated increases much less rapidly with the decrease in the support threshold.

Figure 5(b) shows the execution time of hyperclique miner on the LA1 data set. As can be seen, the execution time reduces significantly with the increase of the h-confidence threshold. Indeed, our algorithm identify hyperclique patterns in just a few seconds at 20% h-confidence threshold and 0.005 support threshold. In contrast, the traditional Apriori (which corresponds to the case that h-confidence is equal to zero in Fig. 5(b)) breaks down at 0.005 support threshold due to excessive memory and computational requirement.

The above results suggest a trade-off between execution time and the number of hyperclique patterns generated at different h-confidence thresholds. In practice, analysts may start with a high h-confidence threshold first at support threshold close to zero, to rapidly extract the strongly affiliated patterns, and then gradually reduce the h-confidence threshold to obtain more patterns that are less tightly-coupled.

Next, we evaluate the performance of hyperclique miner on dense data sets such as Pumsb and Pumsb*. Recently, Zaki and Hsiao proposed the CHARM algorithm (Zaki and Hsiao, 2002) to efficiently discover frequent closed itemsets. As shown in their paper, for the Pumsb and Pumsb* data sets, CHARM can achieve relatively better performance than other state-of-the-art pattern mining algorithms such as CLOSET (Pei et al., 2000) and MAFIA (Burdick et al., 2001) when the support threshold is low. Hence, for the Pumsb and Pumsb* data sets, we chose CHARM as the base line for the case when the h-confidence threshold is equal to zero.

Figure 6(a) shows the number of patterns generated by hyperclique miner and CHARM on the pumsb data set. As can be seen, when the support threshold is low, CHARM can generate a huge number of patterns, which is hard to analyze in real applications. In contrast, the number of patterns generated by hyperclique miner is more manageable. In addition, CHARM is unable to generate patterns when the support threshold is less than or equals

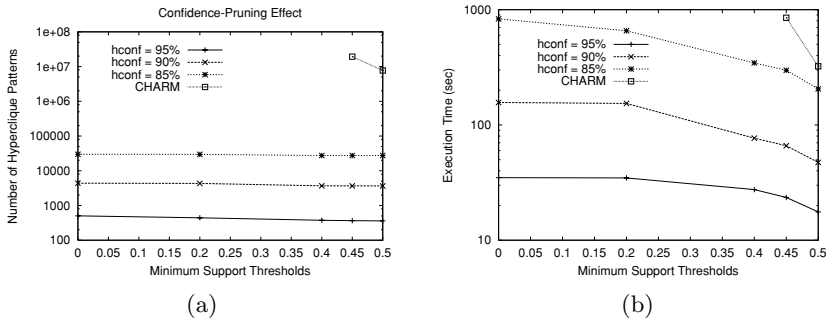


Fig. 6 On the Pumsb data set. (a) Number of patterns generated by hyperclique miner and CHARM. (b) The execution time of hyperclique miner and CHARM

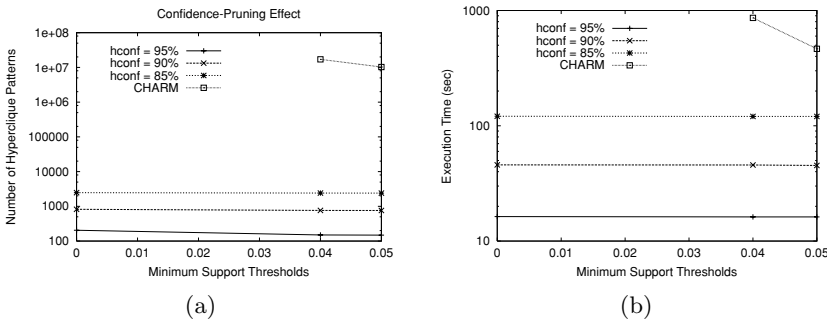


Fig. 7 On the Pumsb* data set. (a) Number of patterns generated by hyperclique miner and CHARM. (b) The execution time of hyperclique miner and CHARM

to 0.4, as it runs out of memory. Recall from Table 1 that nearly 96.6% of the items have support less than 0.4. With a support threshold greater than 0.4, CHARM can only identify associations among a very small fraction of the items. However, hyperclique miner is able to identify many patterns containing items which are strongly related with each other even at very low levels of support. For instance, we obtained a long pattern containing 9 items with the support 0.23 and h-confidence 94.2%. Figure 6(b) shows the execution time of hyperclique miner and CHARM on the pumsb data set. As shown in the figure, the execution time of hyperclique miner increases much less rapidly (especially at higher h-confidence thresholds) than that of CHARM.

Similar results are also obtained from the pumsb* data set, as shown in Figure 7(a) and (b). For the pumsb* data set, CHARM is able to find patterns for the support threshold as low as 0.04. This can be explained by the fact that the pumsb* data set do not include those items having support greater than 0.8, thus manually removing a large number of weakly-related cross-support patterns between the highly frequent items and the less frequent items. Hyperclique miner does not encounter this problem because it automatically removes the weakly-related cross-support patterns using the *cross-support property* of the h-confidence measure. Note that, in the Pumsb* data set, there are still more than 92% (1925 items) of the items that have support less than 0.04. CHARM is unable to find any patterns involving those items with support less than 0.04, since it runs out of memory when the support threshold is less than 0.04.

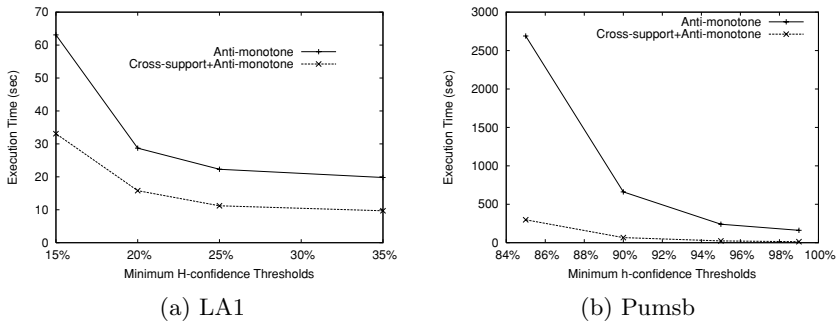


Fig. 8 (a) The effect of cross-support pruning on the LA1 data set. (b) The effect of cross-support pruning on the Pumsb data set

6.3. The effect of cross-support pruning

Figure 8(a) illustrates the effect of cross-support pruning on the LA1 data set. There are two curves in the figure. The lower one shows the execution time when both cross-support and anti-monotone pruning are applied. The above one corresponds to the case that only anti-monotone pruning is applied. As can be seen, cross-support pruning leads to significant reduction in the execution time. Similarly, Fig. 8(b) shows the effectiveness of cross-support pruning on the Pumsb data set. Note that the pruning effect of the cross-support property is more dramatic on the Pumsb data set than on the LA1 data set. This is because cross-support pruning tends to work better on dense data sets with skewed support distributions, such as Pumsb.

6.4. Scalability with respect to number of items

We tested the scalability of hyperclique miner with respect to the number of items on the synthetic data sets listed in Table 3. In this experiment, we set the support threshold to 0.01% and increase the number of items from 1000 to 40000. Figure 9(a) shows the scalability of hyperclique miner in terms of the number of patterns identified. As can be seen, without h-confidence pruning, the number of patterns increases dramatically and the algorithm breaks down for the data set with 40000 items. With h-confidence pruning, the number of patterns generated is more manageable and does not grow fast. In addition, Fig. 9(b) shows the execution time for our scale-up experiments. As can be seen, without h-confidence pruning, the execution time grows sharply as the number of items increases. However, this growth with respect to the number of items is much more moderate when the h-confidence threshold is increased.

6.5. Quality of hyperclique patterns

In this experiment, we examined the quality of patterns extracted by hyperclique miner.

Table 5 shows several interesting hyperclique patterns identified at low levels of support from the LA1 data set. It can be immediately seen that the hyperclique patterns contain words that are closely related to each other. For example, the pattern {arafat, yasser, PLO, Palestine} includes words that are frequently found in news articles about Palestine. These

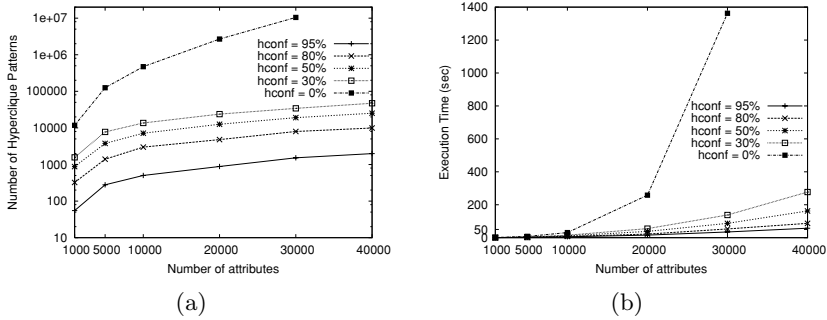


Fig. 9 With increasing number of items. (a) Number of patterns generated by hyperclique miner. (b) The execution time of hyperclique miner

patterns cannot be directly discovered using standard frequent pattern mining algorithms due to their low support values.

Table 6 shows some of the interesting hyperclique patterns extracted at low levels of support from the `retail` data set. For example, we identified a hyperclique pattern involving closely related items such as Nokia battery, Nokia adapter, and Nokia wireless phone. We also discovered several interesting patterns containing very low support items such as {earrings, gold ring, bracelet}. These items are rarely bought by customers, but they are interesting because they are expensive and belong to the same product category.

We also evaluated the affinity of hyperclique patterns by the correlation measure. Specifically, for each hyperclique pattern $X = \{x_1, x_2, \dots, x_k\}$, we calculate the correlation for each pair of items (x_i, x_j) within the pattern. The overall correlation of a hyperclique pattern is then defined as the average pair wise correlation of all the items. Note that this experiment was conducted on the Retail data set with the h-confidence threshold 0.8 and the support threshold 0.0005.

Table 5 Hyperclique patterns from LA1

Hyperclique patterns	support	h-confidence (%)
{najibullah, kabul, afghan}	0.002	54.5
{steak, dessert, salad, sauce}	0.001	40.0
{arafat, yasser, PLO, Palestine}	0.004	52.0
{shamir, yitzhak, jerusalem, gaza}	0.002	42.9
{amal, militia, hezbollah, syrian, beirut}	0.001	40.0

Table 6 Hyperclique patterns from retail

Hyperclique patterns	support	hconf (%)
{earrings, gold ring, bracelet}	0.00019	45.8
{nokia battery, nokia adapter, nokia wireless phone}	0.00049	52.8
{coffee maker, can opener, toaster}	0.00014	61.5
{baby bumper pad, diaper stacker, baby crib sheet}	0.00028	72.7
{skirt tub, 3pc bath set, shower curtain}	0.0026	74.4
{jar cookie, canisters 3pc, box bread, soup tureen, goblets 8pc}	0.00012	77.8

Fig. 10 The average correlation of each pair of items for hyperclique and non-hyperclique patterns

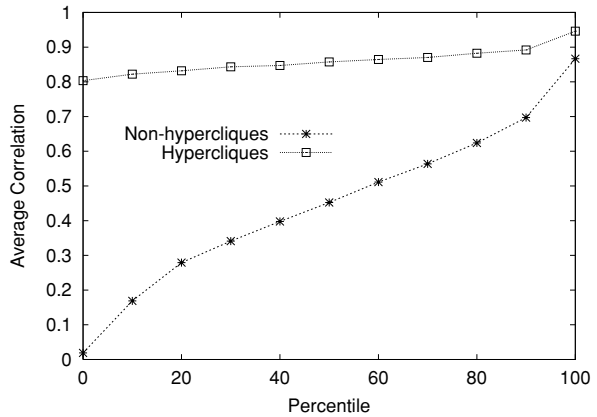


Figure 10 compares the average correlation for hyperclique patterns versus non-hyperclique patterns. We sorted the average correlation and displayed them in increasing order. Notice that the hyperclique patterns have extremely high average pair wise correlation compared to the non-hyperclique patterns. This result empirically shows that hyperclique patterns contain items that are strongly related with each other, especially when the h-confidence threshold is relatively high.

7. Conclusions

In this paper, we formalized the problem of mining hyperclique patterns. We first introduced the concept of the cross-support property and showed how this property can be used to avoid generating spurious patterns involving items from different support levels. Then, we demonstrated, both theoretically and empirically, the role of h-confidence as a measure of association. In addition, an algorithm called hyperclique miner was developed to make use of both cross-support and anti-monotone properties of h-confidence for the efficient discovery of hyperclique patterns even at low levels of support.

There are several directions for future work on this topic. First, the hyperclique miner algorithm presented in this paper is based upon the Apriori algorithm. It will be useful to explore implementations based upon other algorithms for mining hyperclique patterns, such as TreeProjection (Agrawal et al., 2000) and FP-growth (Han et al., 2000). Second, it is valuable to investigate the *cross-support* property on some other measures of association. Third, the current hyperclique pattern mining framework is designed for dealing with binary data. The extension of the hyperclique concept to continuous-valued domains will be a challenging task. Finally, it is a very interesting direction to explore more efficient algorithms based on approximate clique partitioning algorithms (Feder and Motwani, 1995).

Acknowledgments This work was partially supported by NSF grant # IIS-0308264, DOE/LLNL W-7045-ENG-48, and by Army High Performance Computing Research Center under the auspices of the Department of the Army, Army Research Laboratory cooperative agreement number DAAD19-01-2-0014. The content of this work does not necessarily reflect the position or policy of the government and no official endorsement should be inferred. Access to computing facilities was provided by the AHPARC and the Minnesota Supercomputing Institute. Finally, we would like to thank Dr. Mohammed J. Zaki for providing us the CHARM code. Also, we would like to thank Dr. Shashi Shekhar, Dr. Ke Wang, and Michael Steinbach for valuable comments.

References

- Agarwal R, Aggarwal C, Prasad V (2000) A tree projection algorithm for generation of frequent itemsets. *Journal of Parallel and Distributed Computing*
- Agrawal R, Imielinski T, Swami A (1993) Mining association rules between sets of items in large databases. In: *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*. pp 207–216
- Agrawal R, Srikant R (1994) Fast algorithms for mining association rules. In: *Proceedings of the 20th International Conference on Very Large Data Bases*
- Bayardo R, Agrawal R, Gunopulous D (1999) Constraint-based rule mining in large, dense databases. In: *Proceedings of the Int'l Conference on Data Engineering*
- Bayardo RJ (1998) Efficiently mining long patterns from databases. In: *Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data*
- Brin S, Motwani R, Silverstein C (1997) Beyond market baskets: Generalizing association rules to correlations. In: *Proceedings of ACM SIGMOD Int'l Conference on Management of Data*. pp 265–276
- Burdick D, Calimlim M, Gehrke J (2001) MAFIA: A maximal frequent itemset algorithm for transactional databases. In: *Proceedings of the 2001 Int'l Conference on Data Engineering (ICDE)*
- Cohen E, Datar M, Fujiwara S, Gionis A, Indyk P, Motwani R, Ullman J, Yang C (2000) Finding interesting associations without support pruning. In: *Proceedings of the 2000 Int'l Conference on Data Engineering (ICDE)*. pp 489–499
- Feder T, Motwani R (1995) Clique partitions, graph compression and speeding-up algorithms. Special Issue for the STOC conference. *Journal of Computer and System Sciences* 51:261–272
- Grahne G, Lakshmanan LVS, Wang X (2000) Efficient mining of constrained correlated sets. In: *Proceedings of the Int'l Conference on Data Engineering*
- Han J, Pei J, Yin Y (2000) Mining frequent patterns without candidate generation. In: *Proceedings of ACM SIGMOD Int'l Conference on Management of Data*
- Hastie T, Tibshirani R, Friedman J (2001) *The elements of statistical learning: Data mining, inference, and prediction*, Springer
- Liu B, Hsu W, Ma Y (1999) Mining association rules with multiple minimum supports. In: *Proceedings of the 1999 ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*
- Omicinski E (2003) Alternative interest measures for mining associations. *IEEE Transactions on Knowledge and Data Engineering* 15(1)
- Pei J, Han J, Mao R (2000) CLOSET: An efficient algorithm for mining frequent closed itemsets. In: *ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*
- Reynolds HT (1977) *The analysis of cross-classifications*. The Free Press
- Rijsbergen CJV (1979) *Information retrieval*, 2nd edn., Butterworths, London
- Tan P, Kumar V, Srivastava J (2002) Selecting the right interestingness measure for association patterns. In: *Proceedings of the 1999 ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*
- Wang K, He Y, Cheung D, Chin Y (2001) Mining confident rules without support requirement. In: *Proceedings of the 2001 ACM International Conference on Information and Knowledge Management (CIKM)*
- Xiong H, He X, Ding C, Zhang Y, Kumar V, Holbrook S (2005) Identification of functional modules in protein complexes via hyperclique pattern discovery. In: *Proceedings of the Pacific Symposium on Biocomputing (PSB)*
- Xiong H, Steinbach M, Tan P-N, Kumbar V (2004) HICAP: Hierarchical clustering with pattern preservation. In: *Proceedings of 2004 SIAM Int'l Conference on Data Mining (SDM)*. pp 279–290
- Xiong H, Tan P, Kumar V (2003a) Mining hyperclique patterns with confidence pruning. In: *Technical Report 03-006*, January, Department of computer science, University of Minnesota, Twin Cities
- Xiong H, Tan P, Kumar V (2003b) Mining strong affinity association patterns in data sets with skewed support distribution. In: *Proceedings of the 3rd IEEE International Conference on Data Mining*. pp 387–394
- Yang C, Fayyad UM, Bradley PS (2001) Efficient discovery of error-tolerant frequent itemsets in high dimensions. In: *Proceedings of the 1999 ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*
- Zaki M, Hsiao C-J (2002) CHARM: An efficient algorithm for closed itemset mining. In: *Proceedings of 2002 SIAM International Conference on Data Mining*
- Zipf G (1949) *Human behavior and principle of least effort: An introduction to human ecology*. Addison Wesley, Cambridge, Massachusetts