# Mining Strong Affinity Association Patterns in Data Sets with Skewed Support Distribution

Hui Xiong *
Computer Science and Engineering
Univ. of Minnesota - Twin Cities
huix@cs.umn.edu

Pang-Ning Tan
Computer Science and Engineering
Michigan State University
ptan@cse.msu.edu

Vipin Kumar
Computer Science and Engineering
Univ. of Minnesota - Twin Cities
kumar@cs.umn.edu

## Abstract

*Existing association-rule mining algorithms often rely on the support-based pruning strategy to prune its combinatorial search space. This strategy is not quite effective for data sets with skewed support distributions because they tend to generate many spurious patterns involving items from different support levels or miss potentially interesting low-support patterns. To overcome these problems, we propose the concept of hyperclique pattern, which uses an objective measure called h-confidence to identify strong affinity patterns. We also introduce the novel concept of cross-support property for eliminating patterns involving items with substantially different support levels. Our experimental results demonstrate the effectiveness of this method for finding patterns in dense data sets even at very low support thresholds, where most of the existing algorithms would break down. Finally, hyperclique patterns also show great promise for clustering items in high dimensional space.*

## 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 appear many times, most of the words appear only a few times. Such a distribution has been observed in other application domains, including retail data, Web click-streams, and telecommunication data.

This paper examines the problem of applying association analysis [1, 2] to data sets with skewed support distributions. Existing algorithms often use a minimum support threshold to prune its combinatorial search space [2]. Two major problems arise when applying such strategy to skewed data sets.

- If the minimum support threshold is too low, many *uninteresting patterns involving items with substantially different support levels* are extracted. (We call such patterns as `cross-support` patterns.) An example of a cross-support pattern is $\{$`Caviar`, `Milk`$\}$, where `Caviar` is a low support item and `Milk` is a high support item. It is not surprising to find `Milk` in transactions that contain `Caviar` since `Milk` is present in many transactions. Cross-support patterns also tend to have very low pairwise correlations [4].
- If the minimum support threshold is too high, many *strong affinity patterns involving items with low support levels* are missed [8]. Such patterns are useful for capturing associations among rare but expensive items such as caviar and vodka or necklaces and earrings.

To illustrate these problems, consider the support distribution for the `pumsb` census data set shown in Figure 1. `Pumsb` is often used as benchmark for evaluating the performance of association rule algorithms on dense data sets. Observe the skewed nature of the support distribution, with 81.5% of the items having support less than 1% while 0.95% of them having support greater than 90%.

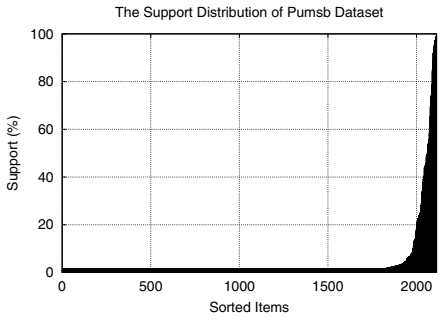


**Figure 1. The support distribution of Pumsb.**

We can partition the items into five disjoint groups based on their support levels, as shown in Table 1. The group

*Contact Author.

Proceedings of the Third IEEE International Conference on Data Mining (ICDM'03)
0-7695-1978-4/03 $ 17.00 © 2003 IEEE

**Table 1. Groups of items for pumsb data set.**

| Group | $S1$ | $S2$ | $S3$ | $S4$ | $S5$ |
|---|---|---|---|---|---|
| Support | 0-1% | 1%-5% | 5%-40% | 40%-90% | >90% |
| # Items | 1735 | 206 | 101 | 51 | 20 |

$S1$ has the lowest support level (less than or equal to 1%) but contains the most number of items, i.e., 1735 items. To detect patterns involving items from $S1$, the minimum support threshold must be less than 1%, but such a low threshold will degrade the performance of existing algorithms considerably. Our experiments showed that state-of-the-art algorithms such as Apriori [2], Charm [16], and MAFIA [5] break down when applied to pumsb at support threshold less than 40% [1]. Furthermore, if the support threshold is low, e.g., 0.05%, many cross-support patterns involving items from both $S1$ (rare items) and $S5$ (very frequent items) are generated. Just to give an indication of the scale, out of the 18847 frequent pairs involving items from $S1$ and $S5$, about 93% of them are cross-support patterns. These cross-support patterns have extremely poor correlation because the presence of the item from $S5$ does not necessarily imply the presence of the item from $S1$. It would be advantageous to develop techniques that can automatically eliminate such patterns during the mining process.

Omiecinski recently proposed an alternative to the support measure called *all-confidence* [10], which represents the minimum confidence of all association rules extracted from an itemset. Omiecinski proved that all-confidence has the desirable anti-monotone property that allows us to incorporate the measure directly into the mining process. We call the patterns derived from this measure as *hyperclique patterns*. (Note that we had independently proposed a measure called h-confidence [15] to capture the degree of affinity in a hyperclique pattern. The equivalence between h-confidence and all-confidence measures is shown in Section 2. For brevity, we will use the term h-confidence when referring to the affinity measure of hyperclique patterns.)

In this paper, we introduce a novel concept called the *cross-support property*, which is useful for eliminating candidate patterns having items with widely differing support levels. We show that h-confidence possesses such a property and develop an efficient algorithm called *hyperclique miner* that utilizes both the cross-support and anti-monotone properties of the h-confidence measure. Our experimental results suggest that hyperclique miner can efficiently discover strong affinity patterns even when the support threshold is set to zero.

Hyperclique patterns are also valuable patterns in their own right because they correspond to itemsets involving only tightly-coupled items. Discovering such patterns can be potentially useful for a variety of applications such as item clustering, copy detection, and collaborative filtering. We demonstrate one potential application of hyperclique patterns in the area of item clustering, where such patterns can be used to provide high-quality hyperedges to seed the hypergraph-based clustering algorithms [7].

**Related Work:** Support-based pruning does not work well with dense data sets, nor is it effective at finding low support patterns. The concepts of maximal [3, 5] and closed itemsets [11, 16] were proposed to address these limitations. Although these concepts can identify a smaller set of representative patterns, their algorithms may still break down at low support thresholds, especially for data sets with skewed support distribution. Both closed and maximal itemsets are also not designed to explicitly remove the cross-support patterns. There has also been growing interest in developing techniques for mining association patterns without support constraints [6, 14]. However, such techniques are either limited to analyzing pairs of items [6] or does not address the cross-support problem [14].

## 2 Hyperclique Pattern

In this section, we describe the concept of a hyperclique pattern and introduce some important properties of the h-confidence measure.

### 2.1 Hyperclique Pattern Concepts

**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 follows from the definition of association rule confidence [1].*

**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 is the support [1] of an itemset. Then* $conf\{A \rightarrow B, C\}$ *= supp({A, B, C})/supp({A}) = 0.6,* $conf\{B \rightarrow A, C\} = 0.6$*, and* $conf\{C \rightarrow A, B\} = 1$*. Hence,* $hconf(P) = min\{conf\{B \rightarrow A, C\}, conf\{A \rightarrow B, C\}, conf\ \{C \rightarrow A, B\}\}$ *= 0.6.*

**Definition 2** *Given a transaction database and the set of all items* $I = \{I_1, I_2, \ldots, I_n\}$*, an itemset* $P$ *is a* **hyperclique pattern** *if and only if: 1)* $P \subseteq I$ *and* $|P| > 0$*. 2)* $hconf(P) \geq h_c$*, where* $h_c$ *is the h-confidence threshold.*

A hyperclique pattern $P$ is a strong-affinity association pattern because the presence of any item $x \in P$ in a transaction strongly implies the presence of $P - \{x\}$ in the same

[1] This is observed on a Sun Ultra 10 workstation with a 440MHz CPU and 128 Mbytes of memory

transaction. To that end, the h-confidence measure is designed specifically for capturing such strong affinity relationships. Nevertheless, hyperclique patterns can also miss some interesting patterns; e.g., an itemset {A, B, C} that produces low confidence rules $A \rightarrow BC$, $B \rightarrow AC$, and $C \rightarrow AB$ but a high confidence rule $AB \rightarrow C$. Such type of patterns are not the focus of this paper.

## 2.2 Properties of h-confidence

We illustrate three important properties of the h-confidence measure in this subsection.

### 2.2.1 Anti-monotone Property

As previously noted, the h-confidence measure is mathematically equivalent to the all-confidence measure proposed by Omiecinski [10], even though both measures are developed from different perspectives.

**Definition 3** *The all-confidence measure [10] for an itemset $P = \{i_1, i_2, \cdots, i_m\}$ is defined as* $\min(\{conf(A \rightarrow B | \forall A, B \subset P, A \cup B = P, A \cap B = \emptyset\})$.

**Lemma 1** *If $P = \{i_1, i_2, \cdots, i_m\}$ is an itemset, then $hconf(P)$ is mathematically equivalent to all-confidence$(P)$ and is equal to*

$$\frac{supp(\{i_1, i_2, \ldots, i_m\})}{max_{1 \leq k \leq m}\{supp(\{i_k\})\}}. \quad (1)$$

Omiecinski proved that all-confidence is an anti-monotone measure [10], i.e., if an itemset $\{i_1, \ldots, i_m\}$ is above the all-confidence threshold, so is every subset of size m-1. Since h-confidence is mathematically identical to all-confidence, it is also monotonically non-increasing as the size of the hyperclique pattern increases. This anti-monotone property allows us to push the h-confidence constraint into the search algorithm. Thus, when searching for hyperclique patterns, the support of a candidate pattern $\{i_1, \ldots, i_m\}$ is counted only if all its subsets of size m-1 are hyperclique patterns.

### 2.2.2 Cross-Support Property

In this section, we introduce the concept of *cross-support* property. This property is useful to *avoid generating cross-support patterns*, which are patterns containing items from substantially different support levels. We also show that the h-confidence measure possesses such a property.

Before presenting the concept of $cross\text{-}support$ property, we first introduce the idea of an upper bound function for a measure of association.

**Definition 4** *Let $f$ be a measure of association and $f_{\max}$ be its maximum possible value. We define $upper(f)$ as an upper bound function for $f$ if $\forall P : f(P) \leq upper(f(P))$, where $P$ is an association pattern. An upper bound function is trivial if $upper(f) = f_{\max}$.*

For example $upper(supp(P)) = 1$ is a trivial upper bound function for the support measure. An example of a non-trivial upper bound function for the h-confidence measure is presented below.

**Lemma 2** *Given an itemset $P = \{i_1, \ldots, i_m\}$, the h-confidence for $P$ has the following upper bound:*

$$upper(hconf(P)) = \frac{min_{1 \leq l \leq m}\{supp(\{i_l\})\}}{max_{1 \leq k \leq m}\{supp(\{i_k\})\}}. \quad (2)$$

We will use the notion of upper bound function to describe cross-support property. In a nutshell, given a specified threshold t, if a function $f$ has the cross-support property, we can find two itemsets from different support levels such that, for any $cross\text{-}support$ pattern $P$, we are guaranteed to have $f(P) < t$. The formal definition of the *cross-support* property of a function $f$ is given below.

**Definition 5** *Let $I = \{i_1, i_2, \cdots, i_n\}$ be an ordered set of items, sorted according to their support values, i.e., $\forall k : supp(i_k) \leq supp(i_{k+1})$. In addition, for each item $x \in I$, let $L(x) = \{x' | supp(x') \leq supp(x)\}$ and $U(x) = \{x' | supp(x') \geq supp(x)\}$.*

*A function $f$ satisfies the cross-support property if $\exists x, y \in I$ such that $supp(x) < supp(y)$ and upper $(f(x, y)) < t$ implies $\forall P : f(P) < t$, where $P$ is an itemset containing at least one item from $L(x)$ and at least one item from $U(y)$ and $t$ is the specified threshold.*

In the following, we provide a sufficient condition for $f$ to satisfy the cross-support property.

**Theorem 1** *Given: 1) A measure of association, $f$; 2) A pair of items $x$ and $y$ with $supp(x) < supp(y)$; 3) A pair of itemsets $L(x) = \{x' | supp(x') \leq supp(x)\}$ and $U(y) = \{y' | supp(y') \geq supp(y)\}$;*

*If the following conditions hold,*

*1) A non-trivial upper bound function for $f$ exists;*

*2) $upper(f(\{x, y\}))$ is computed using only $supp(x)$ and $supp(y)$;*

*3) $upper(f(\{x, y\}))$ decreases monotonically with increasing $supp(y)$ if $x$ is fixed;*

*4) $upper(f(\{x, y\}))$ decreases monotonically with decreasing $supp(x)$ if $y$ is fixed;*

*5) $f$ is an anti-monotone measure when applied to patterns of size two or more;*

*Then* $f(p) \leq upper(f(\{x,y\}))$*, where* $p$ *is a cross-support pattern that contains at least one item from* $L(x)$ *and at least one item from* $U(y)$*.*

The proof of this theorem is given in [15]. As a consequence, $upper(f(\{x,y\})) < t$ implies $f(p) < t$, which means that $f$ must satisfy the cross-support property.

**Lemma 3** *The h-confidence measure satisfies the* $cross\text{-}support$ *property. Furthermore, the h-confidence value for any cross-support pattern* $P = \{x_{i_1}, x_{i_2}, \ldots, x_{i_k}, y_{j_1}, y_{j_2}, \ldots, y_{j_l}\}$ *has an upper bound as* $\frac{max_{1 \leq p \leq m}\{supp(\{x_p\})\}}{min_{1 \leq q \leq n}\{supp(\{y_q\})\}}$

Lemma 3 provides an upper bound of the h-confidence values for all possible cross-support patterns from two itemsets with different support levels. Thus, if the h-confidence threshold is set higher than this upper bound, we will not generate any cross-support pattern as candidate hyperclique pattern during the mining process.

h-confidence is not the only measure that satisfies the cross-support property. Table 2 provides a list of other measures of association that possess such a property. Among the measures that do not have the $cross\text{-}support$ property include support and odds ratio [13].

**Table 2. Measures with the** $cross\text{-}support$ **property (Assume that** $supp(x) < supp(y)$**).**

| Measure | Computation Formula | Upper Bound |
|---|---|---|
| Cosine | $\frac{supp(x,y)}{\sqrt{supp(x)\,supp(y)}}$ | $\sqrt{\frac{supp(x)}{supp(y)}}$ |
| Jaccard | $\frac{supp(x,y)}{supp(x)+supp(y)-supp(x,y)}$ | $\frac{supp(x)}{supp(y)}$ |
| PS | supp(x, y)-supp(x)supp(y) | supp(x)(1-supp(y)) |

### 2.2.3 Strong Affinity Property

In this subsection, we investigate the relationships between h-confidence and other similarity measures such as cosine (Lemma 4) and Jaccard (Lemma 5) measures. Our goal is to derive the lower bounds for these similarity measures in terms of the h-confidence threshold, $h_c$.

**Definition 6** *Given a pair of items* $P = \{i_1, i_2\}$*, the cosine measure [12] for* $P$ *can be computed as* $\frac{supp(\{i_1,i_2\})}{\sqrt{supp(i_1) \times supp(i_2)}}$*, while the Jaccard measure [12] for* $P$ *is* $\frac{supp(\{i_1,i_2\})}{supp(\{i_1\})+supp(\{i_2\})-supp(\{i_1,i_2\})}$.

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

**Lemma 5** *If* $P = \{i_1, i_2\}$ *is a size-2 hyperclique pattern, then* $Jaccard(P) \geq h_c/2$.

The above lemmas suggest that if $h_c$ is sufficiently high, then all size-2 hyperclique patterns contain items that are strongly affiliated with each other in terms of their cosine and Jaccard values. For a hyperclique pattern that contains more than two items, we can compute the average Jaccard and cosine measure for all pairs of items within this pattern. Due to the antimonotone property of the h-confidence measure, every pair of items within a hyperclique pattern must have an h-confidence value greater than or equal to $h_c$. As a result, the average Jaccard or cosine measure of a hyperclique pattern must also satisfy the above lemmas.

## 3 Hyperclique Miner Algorithm

In this section, we design a level-wise algorithm, called hyperclique miner, for discovering hyperclique patterns.

**Example 2** *We illustrate how hyperclique miner works using the running example shown in Figure 2 . As can be seen, the process of searching hyperclique patterns is illustrated by the generation of branches of a set-enumeration tree. For this running example, suppose the minimum support threshold is zero and the minimum h-confidence threshold is 55%. There are two major pruning techniques incorporated into our algorithm.*

1. *We can prune itemsets by the anti-monotone property of the h-confidence measure. For instance, applying Equation 1, the h-confidence of the candidate pattern* $\{4, 5\}$ *is supp(*$\{4, 5\}$*)/max{supp(* $\{4\}$*), supp(*$\{5\}$*)}= 0.1/0.2 = 0.5, which is less than 55%. Hence, the itemset* $\{4, 5\}$ *is not a hyperclique pattern and is immediately pruned. In turn, we can prune the candidate pattern* $\{3, 4, 5\}$ *by the anti-monotone property of the h-confidence measure since one of its subset,* $\{4, 5\}$*, is not a hyperclique pattern.*

2. *We can do pruning by the cross-support property of h-confidence. For instance, given a sorted list of items,* $\{1, 2, 3, 4, 5\}$*, suppose we split the list into two sets* $S_1$=$\{1, 2\}$ *and* $S_2$=$\{3, 4, 5\}$*. We can compute the upper bound of h-confidence for any cross-support pattern between these two item sets by Lemma 3. In this example, the upper bound is equal to max{supp(*$\{3\}$*), supp(*$\{4\}$*), supp(*$\{5\}$*)}/min{supp (*$\{1\}$*), supp(*$\{2\}$*)} = 3/9= 0.33. Therefore, the h-confidence for every cross-support pattern involving these items must be less than 33%. If the h-confidence threshold is 55%, we may prune all these cross-support patterns even before they are generated as candidate patterns. Without applying cross-support pruning, we have to generate six additional patterns, including* $\{1, 3\}$*,* $\{1, 4\}$*,* $\{1, 5\}$*,* $\{2, 3\}$*,* $\{2, 4\}$*, and* $\{2, 5\}$*, as candidate hyperclique patterns and prune them later upon computing their actual h-confidence values. Note that the anti-monotone*

*property does not help us to pre-prune the six candidate patterns, since every subset of these patterns are hyperclique patterns (according to Equation 1, the h-confidence values of size-1 itemsets are 1).*

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

| Item | Support |
|---|---|
| 1 | 0.9 |
| 2 | 0.9 |
| 3 | 0.3 |
| 4 | 0.2 |
| 5 | 0.2 |

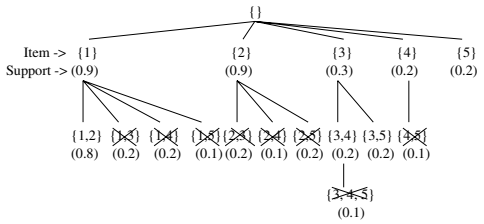


**Figure 2. A running example.**

**Hyperclique Miner**
**Input**:

1) a set F of $K$ Boolean feature types F=$\{f_1, f_2, \ldots, f_K\}$

2) a set T of $N$ transactions T=$\{t_1 \ldots t_N\}$, each $t_i \in T$ is a record with K attributes $\{i_1, i_2, \ldots, i_K\}$ taking values in $\{0, 1\}$, where the $i_p$ is the Boolean value for the feature type $f_p$, for $1 \leq p \leq K$.

3) A user specified h-confidence threshold (*min_conf*)

4) A user specified support threshold (*min_supp*)

**Output**:

hyperclique patterns with h-confidence $> min\_conf$ and support $> min\_supp$

**Method**:

1) Get size-1 prevalent items
2) Construct item sets at different levels of support
3) **for** size of itemsets in (2, 3, $\ldots$ , $K-1$) **do**
4) Generate candidate hyperclique patterns
5) Prune based on the support measure
6) Prune based on the h-confidence measure
7) Generate hyperclique patterns

**Algorithm Description:** The Hyperclique Miner prunes the exponential search space based on the following three conditions: 1) Pruning based on anti-monotone property of h-confidence and support. 2) Pruning based on the upper bound of h-confidence. By Lemma 2, if the upper bound for $hconf(c)$ is less than $h_c$, then $hconf(c)$ must also be less than $h_c$. We can easily compute the upper bound of the h-confidence for any candidate itemset since the support for every individual item is stored in memory, 3) Pruning by the $cross\text{-}support$ property of h-confidence.

## 4 Hyperclique-based Item Clustering

This section describes how to use hyperclique patterns for clustering items in high dimensional space. For high dimensional data, traditional clustering schemes such as K-means [9] tend to produce poor results when directly applied to large, high-dimensional data sets. One promising approach proposed by Han et al. [7] is to cluster the data using a hypergraph partitioning algorithm. More specifically, a hypergraph is constructed with individual items as vertices and frequent itemsets as hyperedges connecting between these vertices. For example, if {A, B, C} is a frequent itemset, then a hyperedge connecting the vertices for A, B, and C will be added. The weight of the hyperedge is given by the average confidence of all association rules generated from the corresponding itemset. The resulting hypergraph is then partitioned using a hypergraph partitioning algorithm such as HMETIS (http://www.cs.umn.edu/∼karypis/metis/hmetis/index.html) to obtain clusters.

Although the hypergraph-based clustering algorithm has produced promising results [7], it can be further improved if the initial hypergraph contains a good representative set of high-quality hyperedges. Frequent itemsets may not provide such a good representation because they include cross-support patterns, which may have low affinity but relatively high average confidence. In addition, many low support items cannot be covered by frequent itemsets unless the minimum support threshold is sufficiently low. However, if the threshold is indeed low enough, a large number of frequent itemsets will be extracted, thus resulting in a very dense hypergraph. It will be difficult for a hypergraph partitioning algorithm to partition such a dense hypergraph, which often leads to poor clustering results.

In this paper, we use hyperclique patterns as an alternative to frequent itemsets. In the hypergraph model, each hyperclique pattern is represented by a hyperedge whose weight is equal to the h-confidence of the hyperclique pattern. For example, if $\{A, B, C\}$ is a hyperclique pattern with the h-confidence equals to 0.8, then the hypergraph contains a hyperedge that connects the vertices $A$, $B$, and $C$. The weight for this hyperedge is 0.8.

There are several advantages of using the hyperclique-based clustering algorithm. First, since hyperclique patterns are strong affinity patterns, they can provide a good representative set of hyperedges to seed a hypergraph-based clustering algorithm. Second, hyperclique patterns can be extracted for very low support items without making the hypergraph becomes too dense. Finally, hyperclique-based

clustering algorithm is also more tolerant to noise compared to traditional clustering algorithms such as k-means because it can explicitly remove the weakly related items.

## 5 Experimental Evaluation

For evaluation purposes, we have performed our experiments on real data sets obtained from several application domains. The characteristics of these data sets are summarized in Table 3.

**Table 3. Real Data set Characteristics.**

| Data set | #Item | #Record | Avg. Length | Source |
|---|---|---|---|---|
| Pumsb | 2113 | 49046 | 74 | IBM Almaden |
| S&P 500 | 932 | 716 | 75 | Stock Market |
| Retail | 14462 | 57671 | 8 | Retail Store |

The `pumsb` data set corresponds to a binary version of a census data set. `Retail` is a masked data set obtained from a large mail-order company. In addition, the stock market data set contains events representing the price movement of various stocks that belong to the S&P 500 index from January 1994 to October 1996.

All 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. Note that we have implemented hyperclique miner as an extension to the publicly available implementation of the Apriori algorithm by Borgelt (http://fuzzy.cs.uni-magdeburg.de/∼borgelt). As a result, the performance of hyperclique miner is almost equivalent to Apriori when the h-confidence threshold is set to zero.

### 5.1 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. Recently, the CHARM algorithm was proposed by Zaki et al.[16] to efficiently discover frequent closed itemsets. As shown in their paper, for a dense data set with skewed support distribution such as $Pumsb$, CHARM can achieve relatively better performance than other state-of-the-art pattern mining algorithms such as CLOSET [11], and MAFIA [5] when the support threshold is low. Hence, we chose CHARM as the baseline to compare against the performance of hyperclique miner on dense data sets (even though hyperclique miner and CHARM are actually targeted towards different kinds of patterns).

Figure 3 shows the number of patterns generated by hyperclique miner and CHARM on the `pumsb` data set. As can be seen, the number of patterns discovered by our algorithm is several orders of magnitude smaller than the number of patterns found by CHARM provided that the

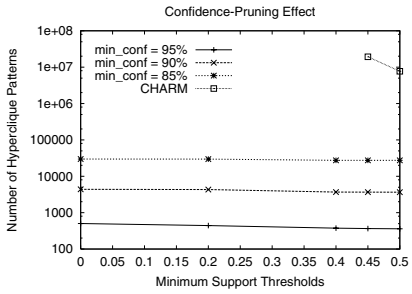


**Figure 3. The effect of h-confidence pruning in terms of the number of patterns generated.**

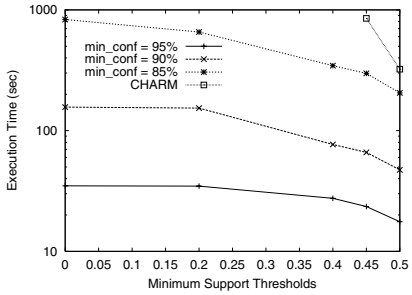


**Figure 4. The effect of h-confidence pruning in terms of execution time.**

h-confidence threshold is sufficiently high. In addition, CHARM has difficulties in identifying patterns when the support threshold is less than or equals to 0.4. However, our technique identifies many strong affinity patterns with very low support. For instance, we obtain a long pattern containing 9 items with the support 0.23 and h-confidence 94.2%. 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. Figure 4 shows the relatively performance of hyperclique miner and CHARM on `pumsb` data set. With h-confidence pruning, we can use hyperclique miner to identify hyperclique patterns even at support threshold equal to zero.

### 5.2 Quality of Hyperclique Patterns

Table 4 shows some of the interesting hyperclique patterns extracted from the `retail` data set. For example, we identified a hyperclique pattern involving closely related items such as Nokia battery, Nokia adapter, and Nokia cell phone. We also discovered several interesting patterns containing very low support items such as {earrings, gold ring, bracelet}. These items are expensive, rarely bought by customers, and belong to the same product category.

**Table 4. Hyperclique Patterns from Retail.**

| Hyperclique patterns | support | h-conf |
|---|---|---|
| {earrings, gold ring, bracelet} | 0.019% | 45.8% |
| {nokia battery, nokia adapter, nokia cell phone} | 0.049% | 52.8% |
| {coffee maker, can opener, toaster} | 0.014% | 61.5% |
| {baby bumper pad, diaper stacker, baby crib sheet} | 0.028% | 72.7% |
| {skirt tub, 3pc bath set, shower curtain} | 0.26% | 74.4% |

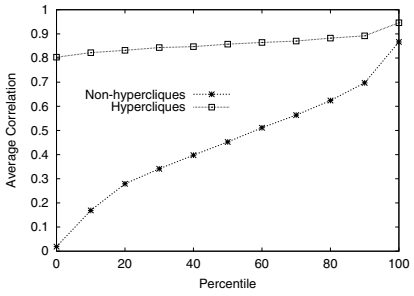


**Figure 5. Average Correlation.**

We also evaluated the affinity of hyperclique patterns by the correlation measure. Specifically, for each hyperclique pattern $X = \{x_1, x_2, \cdots x_k\}$, 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. Note that this experiment was conducted on Retail data set with the h-confidence threshold 0.8 and the support threshold 0.0005.

Figure 5 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 supports our previous assertion that hyperclique patterns can identify itemsets that contain only tightly-coupled items.

### 5.3 Hyperclique-based Item Clustering

In this section, we illustrate the application of hyperclique patterns as an alternative to frequent patterns in hypergraph-based clustering approach [7]. We use the S&P 500 index data set for our clustering experiments.

Table 5 shows the dramatic increase in the number of frequent patterns as the minimum support threshold is decreased. As can be seen, the number of frequent patterns increases up to 11,486,914 when we reduce the support threshold to 1%. If all these frequent itemsets are used for hypergraph clustering, this will create an extremely dense hypergraph and makes the hypergraph-based clustering algorithm becomes computationally intractable. In [7], the authors have used a higher minimum support threshold, i.e., 3%, for their experiments and obtained 19,602 frequent itemsets covering 440 items. A hypergraph consisting of 440 vertices and 19,602 hyperedges was then constructed and 40 partitions were generated. Out of 40 partitions, 16 of them are clean clusters as they contain stocks primarily from the same or closely related industry groups.

**Table 5. Number of frequent patterns.**

| Support | No. of frequent patterns | items covered |
|---|---|---|
| 3% | 19602 | 440 |
| 2% | 149215 | 734 |
| 1% | 11486914 | 915 |

With hyperclique patterns, we can construct hypergraphs at any support threshold, and thus covering more items. For instance, with a minimum h-confidence threshold 20% and a support threshold 0%, we obtain 11,207 hyperclique patterns covering 861 items. A hypergraph consisting of 861 vertices and 11,207 hyperedges is then constructed and partitioned into smaller clusters. For comparison purposes, we partitioned the hypergraph into 80 partitions to ensure that the average size of clusters is almost the same as the average size of the 40 clusters obtained using frequent patterns. Note that for both approaches, we only use patterns containing two or more items as hyperedges.

Our experimental results suggest that the hyperclique pattern approach can systematically produce better clustering results than the frequent pattern approach. First, many items with low levels of support are not included in the frequent pattern approach. Specifically, there are 421 items covered by hyperclique pattern based clusters that are not covered by frequent pattern based clusters. Second, the hypergraph clustering algorithm can produce a larger fraction of clean clusters using hyperclique patterns than frequent patterns — 41 out of 80 partitions versus 16 out of 40 partitions. Third, all the clean clusters identified by the frequent pattern approach were also present in the results by the hyperclique pattern approach. Finally, for the same clean cluster identified by both approaches, there are more same category items included by the hyperclique based approach.

Table 6 shows some of the clean hyperclique pattern based clusters that appear at low levels of support (around 1% support). Such clusters could not be identified by the frequent pattern approach. As the table shows, our hyperclique pattern approach was able to discover retail, chemical, , health-product, power, and communication clusters. A complete list of clusters is given in Technical Report [15].

We have also applied the graph-partitioning scheme in CLUTO [2]. This algorithm takes the adjacency matrix of the similarity graph between the n objects to be clustered as input. The experiment results indicate that this approach can produce much worse clustering results than the hyperclique-based approach. For instance, out of the 80 clusters derived

[2] http://www.cs.umn.edu/∼karypis/cluto/index.html.

**Table 6. Some clean clusters**

| No | Discovered Clusters | Industry Group |
|---|---|---|
| 1 | Becton Dickinso↓, Emerson Electric↓, Amer Home Product↓, Johnson & Johnson↓, Merck↓, Pfizer↓, Schering-Plough↓, Warner-Lambert↓ | health product |
| 2 | duPont (EI) deNemo↑, Goodrich (B.F.)↑, Nalco Chemical↑, Rohm & Haas↑, Avon Products↑ | chemical |
| 3 | Federated Dept↑, Gap Inc↑, Nordstrom Inc↑, Pep Boys-Man↑, Sears↑, TJX↑, Walmart↑ | Retail |
| 4 | Bell Atlantic Co↑, BellSouth Corp↑, CPC Intl↑, GTE Corp↑, Ameritech Corp↑, NYNEX Corp↑, Pacific Telesis↑, SBC Communication↑, US West Communication↑ | Comm. |
| 5 | Baltimore Gas↓, CINergy Corp↓, Amer Electric Power↓, Duke Power↓, Consolidated Edi↓, Entergy Corp↓, Genl Public Util↓, Houston Indus↓, PECO Energy↓, Texas Utilities↓ | Power |

by CLUTO, less than 30 of them are clean clusters. This result is not surprising since the graph-partitioning scheme considers only information about pairs of items but not higher order interactions.

In addition, we also applied the improved version of the k-means clustering algorithm in CLUTO. When using cosine as the similarity measure, we were able to identify 36 clean clusters out of 80 clusters, which is worse than the hyperclique pattern approach.

Finally, we observed the following effects of the hyperclique-based clustering approach. If we set the minimum support threshold to 0% and h-confidence threshold to 20%, the discovered hyperclique patterns cover 861 items. Since there are 932 items in total, the hyperclique pattern mining algorithm must have eliminated 71 items. We examine the distribution of these 71 items in the CLUTO k-means clustering results. We observe that 68 of the items are assigned to the wrong clusters by CLUTO. As a result, we believe that the items not covered by these hyperclique patterns are potentially noise items.

## 6 Conclusions

In this paper, we formalized the problem of mining hyperclique patterns in data sets with skewed support distribution. We also introduced the concept of cross-support property and showed how this property can be used to avoid generating spurious patterns involving items from different support levels. Furthermore, a new algorithm called hyperclique miner was developed. This algorithm utilizes cross-support and anti-monotone properties of h-confidence for the efficient discovery of hyperclique patterns. Finally, we demonstrated applications of hyperclique patterns for discovering strong affinity patterns among low-support items and for hyperclique-based item clustering.

For future work, there is a potential for using the hyperclique concept in a variety of applications such as dimensionality reduction, copy detection, and collaborative filtering. Also, it is valuable to exploit the $cross\text{-}support$ property on some other interestingness measures.

**Acknowledgments** This work was partially supported by NASA grant # NCC 2 1231, NSF grant # ACI-9982274, DOE contract # DOE/LLNL W-7045-ENG-48 and by Army High Performance Computing Research Center contract number DAAD19-01-2-0014. Also, we would like to thank Dr Mohammed J. Zaki for providing us the CHARM code and Dr. Johannes Gehrke for providing us the MAFIA code. Finally, we would like to thank Dr Shashi Shekhar, Dr Ke Wang, and Mr. Michael Steinbach for valuable comments.

## References

[1] R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In *Proc. of the ACM SIGMOD*, pages 207–216, May 1993.

[2] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proc. of the 20th VLDB*, 1994.

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

[4] S. Brin, R. Motwani, and C. Silverstein. Beyond market baskets: Generalizing association rules to correlations. In *Proc. of the ACM SIGMOD*, pages 265–276, 1997.

[5] D. Burdick, M. Calimlim, and J. Gehrke. Mafia: A maximal frequent itemset algorithm for transactional databases. In *Proc. of the ICDE*, 2001.

[6] E. Cohen, M. Datar, S. Fujiwara, A. Gionis, P. Indyk, R. Motwani, J. Ullman, and C. Yang. Finding interesting associations without support pruning. In *ICDE*, 2000.

[7] E. Han, G. Karypis, and V. Kumar. Hypergraph based clustering in high-dimensional data sets: A summary of results. *Bulletin of the Technical Committee on Data Engineering*, 21(1), March 1998.

[8] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2001.

[9] A. Jain and R. Dubes. *Algorithms for Clustering Data*. Prentice Hall, 1998.

[10] E. Omiecinski. Alternative interest measures for mining associations. In *IEEE TKDE*, Jan/Feb 2003.

[11] J. Pei, J. Han, and R. Mao. Closet: An efficient algorithm for mining frequent closed itemsets. In *DMKD*, May 2000.

[12] C. J. V. Rijsbergen. *Information Retrieval (2nd Edition)*. Butterworths, London, 1979.

[13] P. Tan, V. Kumar, and J. Srivastava. Selecting the right interestingness measure for association patterns. In *Proc of the Eighth ACM SIGKDD*, 2002.

[14] K. Wang, Y. He, D. Cheung, and Y. Chin. Mining confident rules without support requirement. In *ACM CIKM*, 2001.

[15] H. Xiong, P. Tan, and V. Kumar. Mining hyperclique patterns with confidence pruning. In *Technical Report 03-006, Computer Science, Univ. of Minnesota.*, Jan 2003.

[16] M. Zaki and C.-J. Hsiao. Charm: An efficient algorithm for closed itemset mining. In *Proc. of the 2nd SDM*, 2002.