Incorporating heuristics for efficient search space pruning in frequent itemset mining strategies

B. Kalpana[1,*] and R. Nadarajan[2]

[1]Department of Computer Science, Avinashilingam University for Women, Coimbatore 641 043, India
[2]Department of Mathematics and Computer Applications, PSG College of Technology, Coimbatore 641 004, India

Recent studies have shown an increasing thrust on the development of algorithms based on a lattice framework. Efficient pruning of the search space is an important factor which determines the performance of such algorithms. In this communication, we present certain lattice theoretical concepts relevant to our work and propose two novel hybrid strategies for enumerating frequent itemsets. Inherent lattice properties along with intelligent heuristics make our algorithms outperform similar algorithms by reducing the search space by almost 50%. We prove theoretically and experimentally that intelligent heuristics applied optimally in alternating the top-down and bottom-up phases results in substantial reduction of the search space.

Keywords: Association rules, data mining, frequent itemsets, heuristics, search space.

FREQUENT itemset mining is one of the fundamental problems in data mining, having varied applications in areas like inductive databases, market basket analysis and stock market predictions, to name a few.

The frequent set counting (FSC) problem as it is known consists of finding all the sets of items which occur in at least *S*% of the transactions of a database *D*, where each transaction is a variable length collection of items from a set *I*. *S* is also referred to as minimum support. Itemsets which have frequency higher than the minimum support are termed frequent. The complexity of the FSC problem is compounded by the exponential growth of its search space, whose dimension *d* in the worst case is given by

$$\sum_{k=1}^{|t_{\max}|} \binom{|I|}{k},$$

where $t_{\max}$ is the maximum transaction length[1]. Based on the minimum support threshold download closure property has been used in many of the previous approaches[2–5] to reduce the search space. But still, a number of computations go wasted before infrequent combinations are known. Our proposed strategies make optimal use of the upward and downward closure properties. This, along with optimal heuristics help in pruning the search space considerably, as we show in the later sections.

*For correspondence. (e-mail: kalpanabsekar@yahoo.com)

Existing algorithms can be broadly classified as apriori-based[2,3], pattern-growth based[6] and lattice-based strategies[5,7]. Such strategies have been used either for determining frequent itemsets, maximal frequent itemsets or closed frequent itemsets. In addition, a number of hybrid approaches which combine several interesting features from different algorithms have been proposed[1,6,8,9].

In the work of Lucchese *et al.*[1], the hybrid nature stems from the ability to switch between a horizontal and vertical database format and the choice of optimized data structures. Uno *et al.*[9] use diffsets or horizontal format based on the density of the input. The hybrid approach of Uno *et al.*[6] is based on the choice of data structures appropriate to the problem. Here we compare the proposed strategies to the hybrid approach by Zaki[5], since we use a combination of bottom-up and top-down procedures to optimally prune the search space. We show theoretically and experimentally that our strategies score over the Maxeclat, which combines a depth first and breadth first search and the Eclat, which uses a pure bottom-up approach. Our techniques use heuristics to cleverly alternate between top-down and bottom-up phases and therefore differ from the previous approaches. The proposed strategies are also different from the Pincer search[10], in that, we use appropriate heuristics to optimally alternate the bottom-up and top-down phases. The objective of the top-down and bottom-up phases also differs from the Pincer search. We give a comparison of our work with the Eclat alone, since it has been established that the Eclat out-performs the Pincer search and is one of the best-perfoming algorithms in its category.

The association mining task, introduced in Agrawal *et al.*[2] can be stated as follows: Given a set of transactions, where each transaction is a set of items, an association rule is an expression $X \Rightarrow Y$, where *X* and *Y* are sets of items. The meaning of such a rule is that transactions in the database which contain the items in *X* also tend to contain the items in *Y*. Two measures which determine the most interesting factors of such a rule are support and confidence. For a given rule expressed as

Bread $\Rightarrow$ Cheese [support = 5%, confidence = 90%],

the measure 'support = 5%' indicates that 5% of all transactions under consideration shows that bread and cheese are purchased together. 'Confidence = 90%' indicates that 90% of the customers who purchased bread also purchased cheese. The association rule mining task is a two-step process.

(1) Find all frequent itemsets. This is both computation and I/O intensive. Given *m* items, there can be potentially 2^m frequent itemsets. It constitutes an area where significant research findings have been reported.

(2) Generating confident rules – Rules of the form $X/Y \Rightarrow Y$ where $Y \subset X$ are generated for all frequent itemsets obtained in step I, provided they satisfy the minimum confidence.

Our focus is on the generation of frequent itemsets. Table 1 shows a sample database with six transactions. The frequent itemsets generated at minimum support 50% is shown in Table 2.

The number in brackets indicates the number of transactions in which the itemset occurs. We call an itemset as frequent if it satisfies the minimum support. A frequent itemset is termed maximal frequent if it is not a subset of any other frequent set for a given minimum support. In our example $\{A, B, C, D\}$ is a maximal frequent itemset at minimum support set to 50%. The proposed hybrid strategies aim at finding out the maximal frequent sets and generating its subsets.

Now we review some of the definitions from lattice and representation theory[11]. We propose lemmas I and II which form the basis of our itemset pruning strategy.

Definition I: Let P be a set. A partial order on P is a binary relation $\leq$, such that for all $X, Y, Z \in P$, the relation is: (1) Reflexive: $X \leq X$. (2) Anti-symmetric: $X \leq Y$ and $Y \leq X$, implies $X = Y$. (3) Transitive $X \leq Y$ and $Y \leq Z$, implies $X \leq Z$. The set P with relation $\leq$ is called an ordered set.

Definition II: Let P be a non-empty ordered set. (1) If $X \vee Y$ and $X \wedge Y$ exist for all $X, Y \in P$, then P is called a lattice. (2) If $\vee S$ and $\wedge S$ exist for all $S \subseteq P$, then P is called a complete lattice.

For a set I, given the ordered set $P(I)$, the power set of I is a complete lattice in which join and meet are given by union and intersection respectively.

$$\vee\{A_i/i \in I\} = \bigcup_{i \in I} A_i,$$

$$\wedge\{A_i/I \in I\} = \bigcap_{i \in I} A_i.$$

Table 1. Sample database

Transactions	Items
1	*A, B, C, D*
2	*A, B*
3	*A, B, C, D, E*
4	*A, B, C, D*
5	*A, C, E*
6	*A, B, C*

Table 2. Frequent itemsets

Frequent itemsets	Support (%) (minimum support = 50)
A	100 (6)
B, C, AC, AB	83 (5)
ABC, BC	67 (4)
BCD, D, ACD, ABCD, AD, ABD	50 (3)

The top element of $P(I)$ and the bottom element of $P(I)$ are given by $T = I$ and $\perp = \{\ \}$ respectively. For any $L \subseteq P(I)$, L is called a lattice of sets if it is closed under finite unions and intersections[5], i.e. $(L, \subseteq)$ is a lattice with partial order specified[5] by the subset relation $\subseteq$, $X \vee Y = X \cup Y$ and $X \wedge Y$.

The power set lattice for our sample database $I = \{A, B, C, D, E\}$ shown in Figure 1, constitutes the search space. Maximal frequent sets are indicated by dark circle. Frequent itemsets are grey circles, while infrequent itemsets are plain circle. It has been observed that the set of all frequent itemsets forms a meet semi lattice. For any frequent itemsets X and Y, $X \cap Y$ is also frequent. The infrequent itemsets form a join semi lattice.

Definition III: Let P be an ordered set and $Q \subseteq P$. (1) Q is a down-set (decreasing set and order ideal) if, whenever $x \in Q$, $y \in P$ and $y \leq x$, we have $y \in Q$. (2) Dually, Q is an up-set (increasing set and order filter) if, whenever $x \in Q$, $y \in P$ and $y \geq x$, we have $y \in Q$.

Given an arbitrary subset Q of P and $x \in P$, we define

$$\downarrow Q = \{y \in P/(\exists x \in Q) y \leq x\} \text{ and}$$

$$\uparrow Q = \{y \in P/(\exists x \in Q) y \geq x\},$$

$$\downarrow x = \{y \in P/y \leq x\} \text{ and } \uparrow x = \{y \in \mathrm{P}/y \geq x\}.$$

Lemma 1: For a maximal frequent itemset $Q \subseteq P$, all down-sets $Q1 = \downarrow Q$; $Q1 \subseteq P$ will also be frequent.

This is a consequence of the above definition. Fast enumeration of the frequent itemsets is possible in the bottom-up phase once the first maximal frequent set is detected. Examining only the potentially frequent itemsets avoids unnecessary Tid list intersections.

Lemma 2: For a minimal infrequent set $Q \subseteq P$ all up-sets $Q1 = \uparrow Q$; $Q1 \subseteq P$ will be infrequent.

The top-down phase detects the minimal infrequent sets. In the power set lattice shown in Figure 1 *AE* is infrequent and it is observed that all up-sets $Q1 = \uparrow Q$ leading to the top element are also infrequent. Both the algorithms alternate the phases in the search heuristically based on the detection of down-sets and up-sets.

Here we propose two algorithms, Hybrid Miner I and Hybrid Miner II, which operate on a vertical database format shown in Figure 2. Accommodating the power set lattice in primary memory is not possible for large datasets since the lattice search space grows exponentially with the items. We use a recursive prefix-based decomposition of the lattice. The tid lists for the items are generated in the first scan of the database. Support is computed by intersection of the tid list for the items constituting the itemset.

For example, in order to find the support of an item set *ABC*, we first perform an intersection of the lists for *A*, *B*

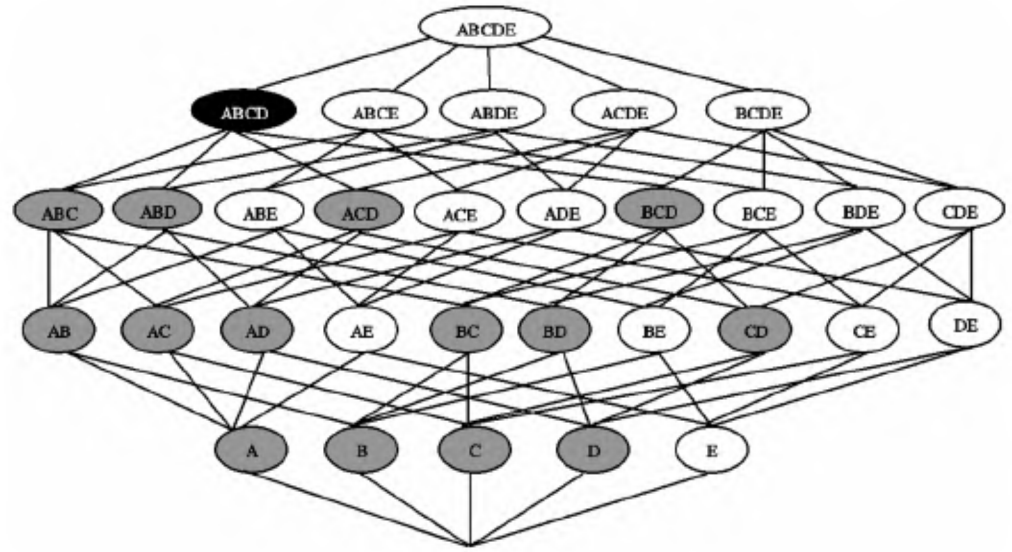

Figure 1. Power set lattice $P(I)$.

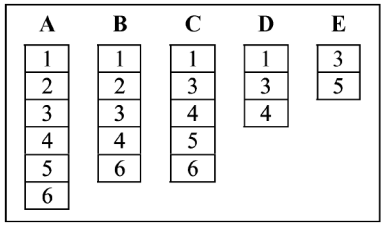

Figure 2. Tid lists for items in sample database.

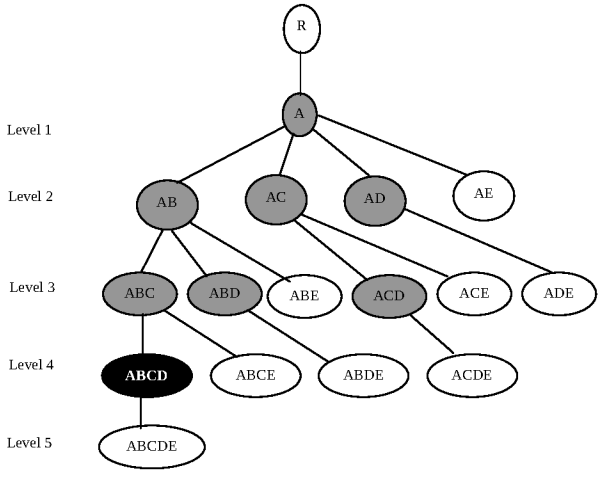

Figure 3. Equivalence class for item A.

and B, C. The cardinality of the set $AB \cap BC$ then gives the support for ABC.

In the bottom-up phase, intermediate computations are temporarily stored and used in the support computation of the next node at the same level. Figure 3 shows the equivalence class corresponding to item A. Figure 4 a and b gives the pseudocode for Hybrid Miner I and Hybrid Miner II respectively.

In the case of Hybrid Miner I, the search starts with a bottom-up phase to identify the maximal frequent item sets. θ indicates the length of the common prefix for decomposing the power set lattice into sub-lattices. It starts at level n – the highest level in a sub-lattice and performs a breadth first search moving to the next lower level if no maximal frequent itemsets are found at the current level. Once the first maximal frequent itemsets at a level are found, we determine items missing from the maximal frequent sets and start a top-down phase that lists the minimal length infrequent sets. The heuristic is gained from the items missing in the maximal frequent sets generated at that level. Faster search is possible because we examine nodes which contain the missing items only. This phase starts at level 2. If no infrequent sets are found at level 2, we go to the next higher level. The top-down phase ends when minimal infrequent sets are detected. The bottom-up phase then resumes to list the other maximal frequent itemsets and frequent items sets after eliminating the nodes containing infrequent itemsets generated in the top-down phase. The computationally intensive support task is thus reduced by cleverly alternating the bottom-up and top-down phases. The process of generating the frequent itemsets is then a simple task of enumerating the subsets of all maximal frequent sets. We also make a check to avoid duplicates. The heuristic here is based on the assumption that the items missing from the maximal frequent itemsets are likely to lead to infrequent combinations. The top-down phase thus examines only potentially infrequent nodes.

(a)

```
/*Bottom-up phase discovers the maximal frequent itemsets,
top-down phase discovers the minimal infrequent itemsets*/
Begin
    Set flag = false;
    for all sub-lattices S induced by θ_k do
    Mfreq = φ;
    Bottom-up (S):
    While there are unmarked nodes
        Repeat until flag = true
            for R ∉ Mfreq
                L(R) = ∩{L(A_i)/A_i ∈ S};  mark node;
            if σ(R) ≥ min_supp then
            Mfreq = Mfreq ∪ R; flag = true;
        top-down (S):
        level = 2 /* starts for 2-length itemsets */
        infreq = φ;
        Repeat for all atoms not in Mfreq
            ∨ A_j ∈ S with j > i do
                L(R) = L(A_i) ∩ L(A_j);
                if σ(R) < min_supp then
                    infreq = infreq ∪ R; break;
                else
                    level = level + 1; continue;
    Repeat Bottom-up for nodes not containing infrequent
    subsets; /* generate freq itemsets */
        freq = { ∪_{i=1}^{Max length} A_i ⊆ Mfreq Λ A_i ⊄ infreq };
end.
```

(b)

```
/*Top-down phase identifies minimal length infrequent itemsets.
Bottom-up phase examines potential nodes only*/
Begin
    for all sub lattices S induced by θ_k do
    /* atoms sorted on ascending order of support */
        top-down (S):
        begin
            level = 2; infreq = φ; flag = false;
            Repeat for all nodes at level while flag = false
              ∨ A_j ∈ S with j > i do
                L(R) = L(A_i) ∩ L(A_j);
                if σ(R) < min_supp then infreq = infreq ∪ R;
                if lastnode and flag = true then break;
                else  level = level + 1;
        end; /*Top-down(S)*/
        Bottom-up (S):
        While there are unmarked nodes
        begin
            Mfreq = φ; level = n;
            for R∉Mfreq and R ∉ Nfreq
                L(R) = ∩{L(A_i) A_i ∈ S}; mark node;
                if σ(R) ≥ min_supp then Mfreq = Mfreq ∪ R;
                else  level = level – 1; continue;
        end;
    freq = { ∪_{i=1}^{Max length} A_i ⊂ Mfreq Λ A_i ⊄ infreq };
end.
```

Figure 4. Pseudocode for (***a***) Hybrid Miner I and (***b***) Hybrid Miner II.

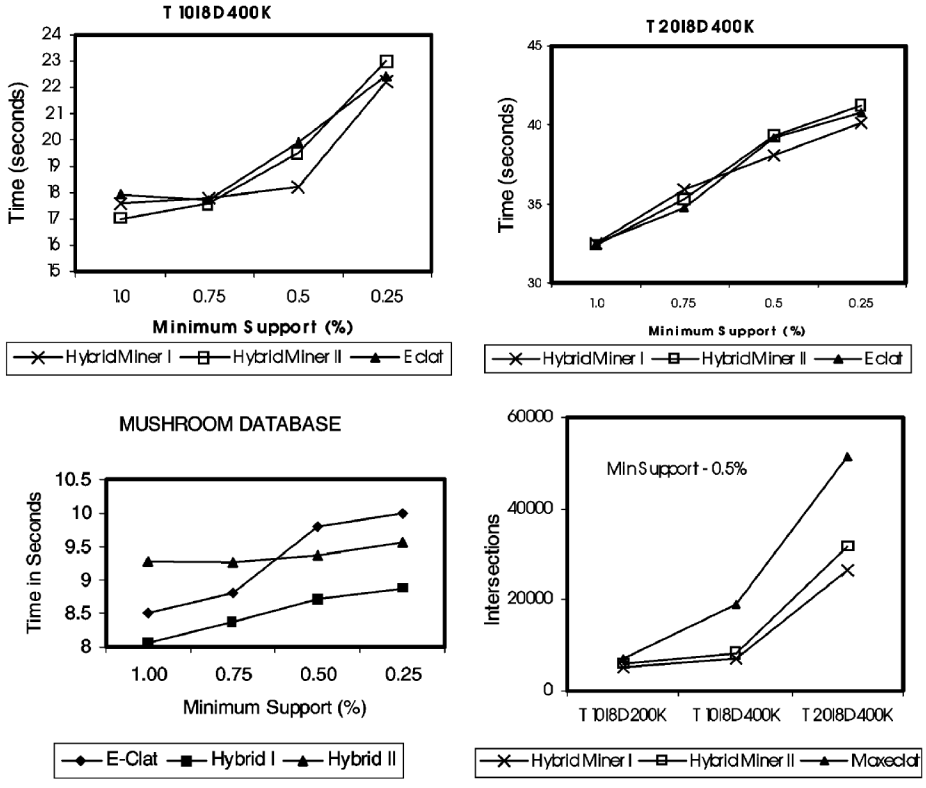

Figure 5. Execution times and Tid list intersections.

Hybrid Miner II starts with a top-down phase to enumerate the minimal length infrequent itemsets. This method examines the nodes in the ascending order of supports. The bottom-up phase starts when minimal length infrequent itemsets are found in an equivalence class. In this phase, the maximal frequent itemsets are generated by only examining nodes not containing the minimal infrequent itemsets. Generating the remaining frequent itemsets is as described for Hybrid Miner I. It is a variation of the Hybrid Miner I, in that it attempts to avoid the intensive computation of supports which are encountered for the candidate nodes in the bottom-up phase in the initial stage itself. Hence efficient subset pruning is incorporated at the start of the algorithm itself. Both the strategies make use of a sentinel routine to take care of any nodes not covered in either of the phases. We now highlight some of the strengths of our algorithms. (i) Significant reduction in I/O and memory. (ii) The sorting of itemsets at second level imposes an implicit ordering on the tree. Each child is attached to the parent with the highest support. Redundancy and overlapping amongst classes is avoided. (iii) On comparison with the approaches in[5], it is found that the number of tid list intersections and nodes examined is reduced by optimally using heuristics to alternate between the top-down and bottom-up phases.

We further draw a theoretical comparison with the best-performing Maxeclat proposed in Zaki[5]. We manually trace the Hybrid Miner I, Hybrid Miner II and Maxeclat for the power set lattice shown in Figure 1. Hybrid Miner I examines only ten nodes to generate the maximal frequent set {*A*, *B*, *C*, *D*} Hybrid Miner II examines 12 nodes, while Maxeclat examines 18 nodes for generating the maximal frequent itemset. Our methods thus achieve a search space reduction of almost 50% over the Maxeclat. The savings in computation time and overhead are significant for large databases.

The experiments were carried out on a Pentium-IV machine with 512 MB RAM, running at 1500 MHz. Synthetic databases were generated using the Linux version of the IBM dataset generator. The data mimic transactions in a retailing environment. The performance of our algorithms is illustrated for the databases T10I8D400K and T20I8D400K. T, I and D indicate the average transaction size, size of a maximal potentially frequent itemset and the number of transactions respectively. The execution times of the proposed algorithms in comparison to Eclat are illustrated in Figure 5. In the case of artificial datasets, Hybrid Miner I performs better than Hybrid Miner II and Eclat for lower supports, whereas Hybrid Miner II performs better for higher supports. For the Mushroom database which is a dense real database, Hybrid Miner I performs better. Figure 5 also shows the tid list-intersections. Both Hybrid Miner I and Hybrid Miner II perform about half the number of intersections compared to Maxeclat. We give a comparison of the tid list intersections only with Maxeclat, since it is a hybrid strategy. Further reduction in time may be possible using more efficient and compressed data structures.

Our experiments have proved that Hybrid Miner I and Hybrid Miner II are efficient search strategies. Both the methods benefit from reduced computations and incorporate excellent pruning that rapidly reduces the search space. Further, both the upward and downward closure properties have been efficiently and optimally utilized. The objective has been to optimize the search for frequent itemsets by applying appropriate heuristics. Further optimizations in terms of storage are being experimented.

1. Lucchese, C., Orlando, S., Palmerini, P., Perego, R. and Silvestri, F., kDCI: A multistrategy algorithm for mining frequent sets. IEEE ICDM Workshop on Frequent Itemset Mining Implementation, 2003.
2. Agrawal, R., Imielinski, T. and Swami, A., Mining association rules between sets of items in large databases. In ACM SIGMOD Conference on Management of Data, May, 1993.
3. Agrawal, R., Mannila, H., Srikant, R., Toivonen, H. and Inkeri Verkamo, A., Fast discovery of association rules. In *Advances in Knowledge Discovery and Data Mining* (ed. Fayyad, U.), AAAI Press, California, USA, 1996, pp. 307–328.
4. Brin, S., Motwani, R., Ullman, J. and Tsur, S., Dynamic itemset counting and implication rules for market basket data. In ACM SIGMOD Conference on Management of Data, June 1997.
5. Zaki, M. J., Scalable algorithms for association mining. *IEEE Trans. Knowledge and Data Eng.*, 2000, **12**, 372–390.
6. Uno, T., Kiyori, M. and Arimura, H., LCM Ver 3: Collaboration of array bitmap and FP tree for frequent itemset mining. In Conference of Knowledge Discovery in Data. Proceedings of First International Workshop on Open Sources Data Mining: Frequent Pattern Mining Implementations, 2005, pp. 77–86.
7. Valtchev, P., Missaoui, R., Godin, R. and Meridji, M., Generating frequent itemsets incrementally: Two novel approaches based on Galois lattice theory. *Comput. Intelligence*, 1999, **15**, 115–142.
8. Uno, T., Asai, T., Uchida, Y. and Arimura, H., LCM: An efficient algorithm for enumerating closed frequent itemsets. In Workshop on Frequent Itemset Mining Implementation (FIMI 03), 19 November 2003, Florida, USA, 2003.
9. Uno, T., Kiyori, M. and Arimura, H., LCM Ver 2: Efficient mining algorithms for frequent closed maximal itemsets. In Workshop on Frequent Itemset Mining Implementation (FIMI 04), 1 November 2004, Brighton, UK, 2004.
10. Lin, D. I. and Kedem, Z. M., Pincer Search – A new algorithm for discovering the maximum frequency set. Sixth International Conference on Extending Database Technology, Spain, March 1998.
11. Davey, B. A. and Priestley, H. A., *Introduction to Lattices and Order*, Cambridge University Press, Cambridge, 1990.

ACKNOWLEDGEMENTS. We thank Dr M. J. Zaki for providing link to the codes of Eclat and Maxeclat for experimental comparison.

Received 16 April 2007; revised accepted 5 November 2007