Optimizing Search Space Pruning in Frequent Itemset Mining With Hybrid Traversal Strategies-A Comparative Performance on Different Data Organizations

B.Kalpana, Dr.R.Nadarajan

Abstract

The task of finding frequent itemsets in a dataset forms the computationally intensive task in association rule mining. The last decade has witnessed a number of state-of-art strategies directed at the Frequent Itemset Mining (FIM) problem. Some of these are hybrid which combine the desirable characteristics of several algorithms. The proposed hybrid strategies employ intelligent heuristics to optimally switch between a bottom up and top down phase to reduce the search space by almost fifty percent.In this paper the performance of the strategies are compared on two dataset organizations.

Index Terms — Association rules, data mining, frequent itemsets, hybrid search.

I.INTRODUCTION

Association rule mining was originally applied in Market Basket Analysis which aims at understanding the behaviour and shopping preferences of retail customers. The knowledge is used in product placement, marketing campaigns and sales promotions.Besides the retail sector, the market basket analysis framework is also being extended to the health and other service sectors. The application of Association rule mining now extend far beyond Market Basket Analysis and includedetection of network intrusion, attacks from the logs of web server and prediciting user traversal patterns on the web.

FIM algorithms could be broadly classified as candidate generation algorithms or pattern growth algorithms. Within these categories further classification can be done based on the traversal strategy and data structures used.Apart from these several hybrid algorithms which combine desirable features of different algorithms have been proposed. Apriori Hybrid, VIPER,Max Eclat, KDCI are some of them. Our work has been motivated by the Eclat and Maxeclat[20] ,which is a hybrid strategy . We propose two hybrid strategies which make an intelligent combination of a bottom up and top down search to rapidly prune the search space.The intelligence gained from each phase is powered to optimally exploit the upward and downward closure properties .The strategies are found to outperform the Eclat and Maxeclat as indicated in section VII. In this paper we give a comparitive performance of the strategies on Tidset and the Diffset organizations. Diffsets[21] have proved to occupy a smaller footprint in the memory and hence are reported to be advantageous.

B.Kalpana is with the Department of Computer Science,Avinashilingam University for Women,Coimbatore,India.
Email:kalpanabsekar@yahoo.com
Dr.R.Nadarajan is with the Department of Mathematics and Computer Applications,PSGCollegeof Technology,Coimbatore,,India
Email:nadarajan_psg@yahoo.co.in

II.PROBLEM STATEMENT

The association mining task, introduced in [1] 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 interestingness 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 show 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 I(a) shows a sample database with six transactions. The frequent itemsets generated at minimum support 50% is shown in Table I(b).

Table I(a) : 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 I(b) : Frequent Itemsets

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

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.

III. CONNECTING LATTICES AND HYBRID SEARCH STRATEGIES

We review some of the definitions from lattice and representation theory [5]. We propose lemma 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.

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

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

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, i.e., $(L, \subseteq)$ is a lattice with partial order specified by the subset relation $\subseteq$, $X \vee Y = X \cup Y$ and $X \wedge Y$ [20].

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

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

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

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

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 powerset lattice shown in fig.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.

IV. ITEMSET ENUMERATION

The enumeration of frequent itemsets forms the computationally intensive task. For a consideration of m distinct items. We can have a combination of 2^m subsets, which results in an exponential growth of the search space. Itemset enumeration research thus focuses on reducing the dataset I/O and containing the exploration. There are four applicable classes of I/O reduction suggested in [1]. They are

i. Projection: The projection of the database onto an equivalent condensed representation reduces storage requirement. It may also result in computational optimization through efficient algorithmic techniques.

ii.Partitioning: Dataset partitioning minimizes I/O costs by enabling memory resident processing of large datasets, thus reducing costly disk accesses.

iii. Pruning: Dataset pruning techniques dynamically reduce the dataset during processing,by discarding unnecessary items. This is significant in reducing the processing time.

iv. Access reduction: Reducing the number of times that disk resident datasets need to be accessed to identify all frequent itemsets.The hybrid strategies that we propose are directed at maximal pruning of the search space by an optimal exploitation of the upward and downward closure properties.

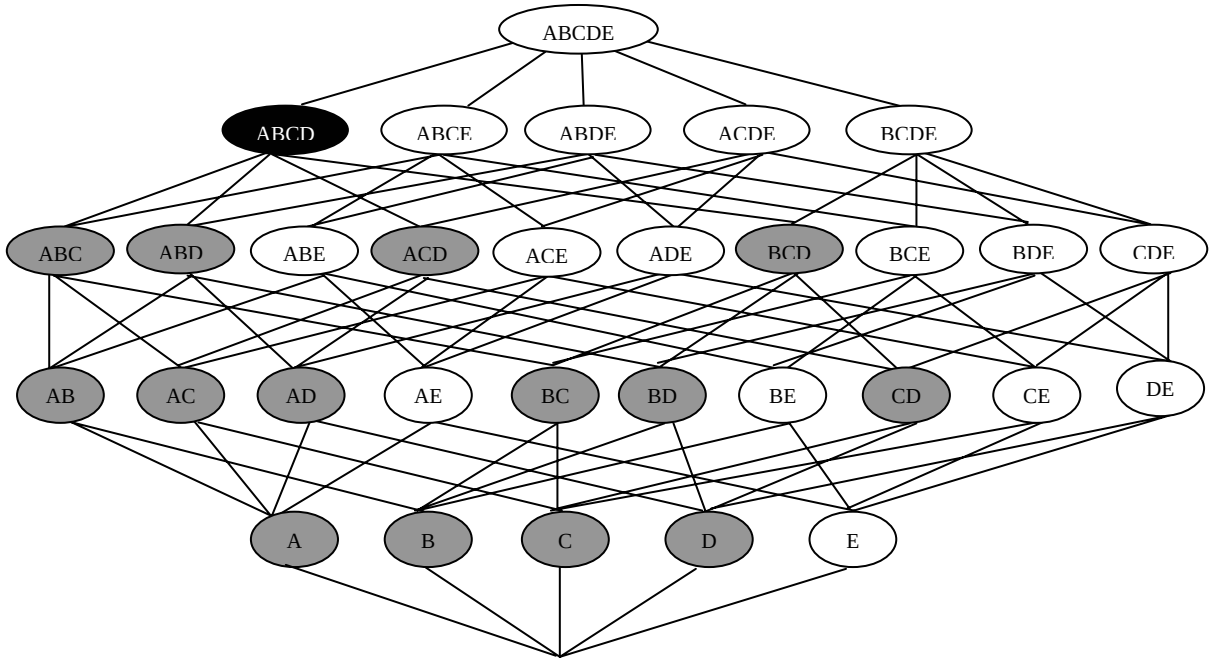
V. DATASET ORGANIZATION

Dataset organizations are typically horizontal or vertical.. In the horizontal format each row contains an object or a transaction id and its related attributes, while in the vertical representation, items are represented as columns each containing the transactions where it occurs.Traditional methods used the horizontal format, whereas some of the recent methds have imcreasingly relied on the vertical format[20] [21] [14]. .Tid sets , diffsets and vertical bit vectors are some of the commonly used vertical data formats.. In [14] compressed vertical bitmaps or snakes were introduced to reduce the vertical representation in comparison to the equivalent horizontal representation. Here we make a comparative study of the performance of two novel hybrid strategies on tidsets and diffsets. In the diffset format we keep track of the differences of the tidlist of an itemset from its generating pattern.Diffsets are reported to reduce the memory requirements and since they are shorter than the tidlists ,the support compuations are faster.

Figure.2 shows the Tidset format and diffset format for the sample database.It is obvious that the diffsets are smaller than the tidlists. In the tidlist format the support for an itemset ABC is computed by intersecting the tidlists of any two of its subsets say AB and BC. The cardinality of the set obtained by this intersection gives the support.The support computation is different in diffsets.The differences in the tidlists of a class member and its prefix itemset is used . The original database is maintained in tidlist format. Support of an itemset ABC is computed recursively as
(ABC) = (AB) - | d(ABC) | applying it recursively, we have, (ABC) = d(AC) - d(AB). For a more detailed explanation one is referred to [21].

VI. ALGORITHM DESIGN & IMPLEMENTATION

We propose two algorithms Hybrid Miner I and Hybrid Miner II which operate on the tidlist format and the diffset format. The pseudocode represents support computation using tidlists only. Support computation using diffsets is explained in the previous section 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 tidlists for the items are genarated in the first scan of the database. Figure 3 shows the equivalence class corresponding to item A.Figures 4 and 5 give the pseudocode for Hybrid Miner I and Hybrid Miner II respectively.



{ } Fig.1.ThePowerset lattice P(I)

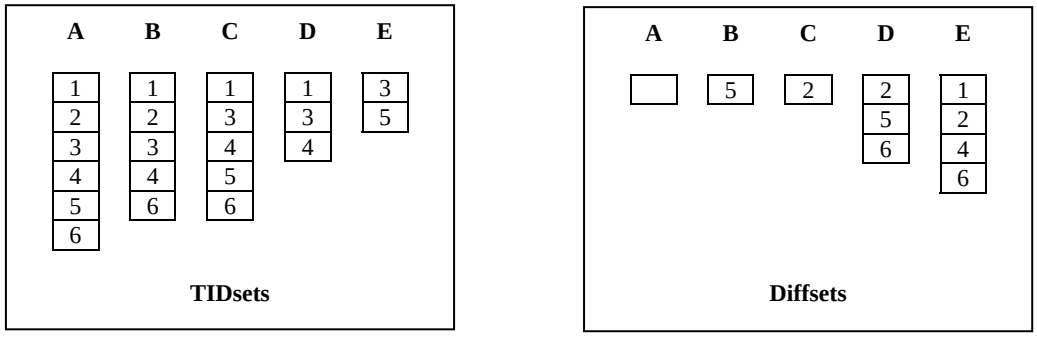


Fig. 2 : TIDSets and DiffSets for Sample Database

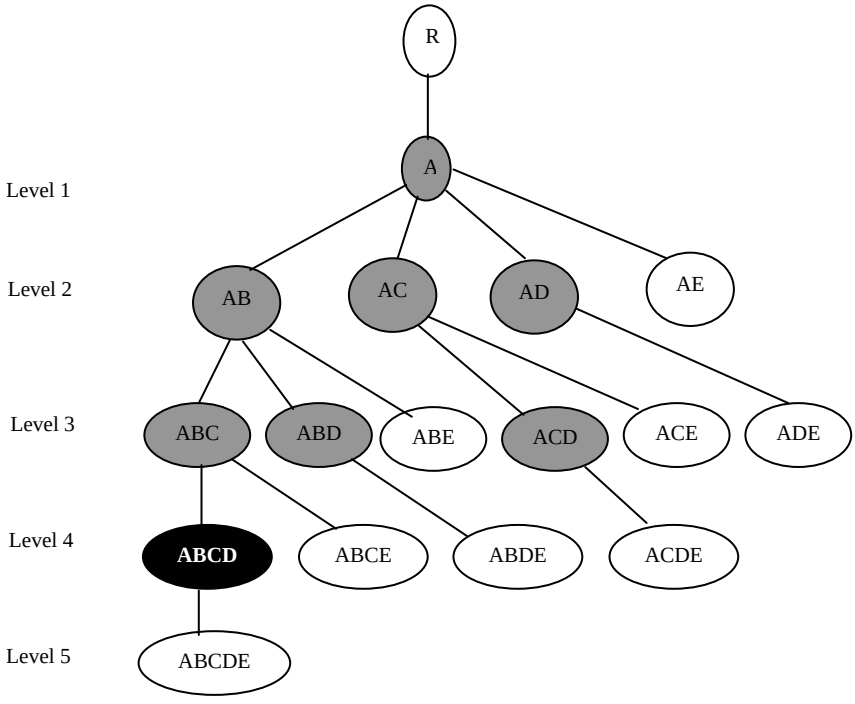


Fig. 3: Equivalence Class for Item A

Hybrid Miner I : /* 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 θk do
    Bottom-up(S):
        Mfreq = ϕ;
        Repeat until flag = true
            for R ∉ Mfreq
                L(R) = ∩{L(Ai) / Ai ∈ S};
            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
            ∨ Aj ∈ S with j > i do
                L(R) = L(Ai) ∩ L(Aj);
                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 = { Max length ∪ (i=1) Ai ⊆ Mfreq Λ Ai ⊄ infreq };

end.
```

Fig. 4.Pseudocode for Hybrid Miner I

Hybrid Miner II /* Top down phase identifies minimal length infrequent itemsets. Bottom up phase examines potential nodes only*/

```
Begin
    for all sub lattices S induced by θk do
    /*  atoms sorted on ascending order of support  */
     topdown (S):
     begin
            level = 2; infreq = ϕ; flag = false;
            Repeat for all nodes at level while flag = false
              ∨ Aj ∈ S with j > i do
                L(R) = L(Ai) ∩ L(Aj);
                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):
        begin
            Mfreq = ϕ; level = n;
            for R∉Mfreq and R ⊄ Nfreq
                L(R) = ∩{L(Ai) / Ai ∈ S};
                if σ(R) ≥ min_supp then MFreq = Mfreq ∪R;
                else   level = level – 1; continue;
     end;
            Max length
        freq = { ∪ (i=1) Ai ⊂ Mfreq Λ Ai ⊄ infreq };
end.
```

Fig. 5.Pseudocode for Hybrid Miner II

A. Description of Hybrid Miner I

The search starts with a bottom up phase to identify the maximal frequent item sets. It starts at level n 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 itemset is found, we determine items missing from the maximal frequent set and start a top down phase that lists the minimal length infrequent sets. 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 computation task is thus reduced by cleverly alternating the bottom up and top down phases everytime a maximal itemset is detected . 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.

B. Description of Hybrid Miner II

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. 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 [20] 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 [20]. We manually trace the Hybrid Miner I, Hybrid Miner II and Maxeclat for the powerset lattice which is shown in Fig. 1. Hybrid Miner I examines only 10 nodes to generate the maximal frequent set {A,B,C,D} Hybrid Miner II examines 12 nodes while Maxeclat will examine 18 nodes for generating the maximal frequent itemset. Our methods thus achieve a search space reduction of almost fifty percent over the Maxeclat. The savings in computation time and overhead is significant for large databases.

VII. EXPERIMENTAL RESULTS

The experiments were carried out on a Pentium-IV machine with 512 MB RAM running at 1500Mhz. Synthetic databases were generated using the Linux version of the IBM dataset generator. The data mimic the transactions in a retailing environment. The performance of our algorithms are illustrated for synthetic and real datasets. T, I and D indicate the average transaction size, the size of a maximal potentially frequent itemset and the number of transactions respectively. . On the Tidlist format the execution times of the proposed algorithms in comparison to Eclat are illustrated in Figure 6. Hybrid Miner I performs better than Hybrid Miner II and Eclat for lower supports whereas Hybrid Miner II performs better for higher supports. Figure 7 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 through more efficient and compressed data structures. Figure 8 shows the performance of the two strategies on the tidset and diffset organizations.The hybrid strategies on the diffset format are advantageous on the dense datasets. T10I8D100k is a relatively sparse dataset.There is no significant advantage here. However on T10I8D400k,T20I8D400K and the mushroom dataset the hybrid strategies benefit from reduced execution times while using the diffset format. The results indicate that the diffset organization may be more suitable for dense datasets. The choice of the traversal strategy may also favour a particular data format. Since the hybrid strategies use a combination of traversal mechanisms, the diffset organization offers only a moderate advantage in terms of execution times as is indicated in the graphs of figure 6 .

VIII. CONCLUSION

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.From the experiments on two different data formats, we find that the diffset format is better in the case of dense datasets 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.

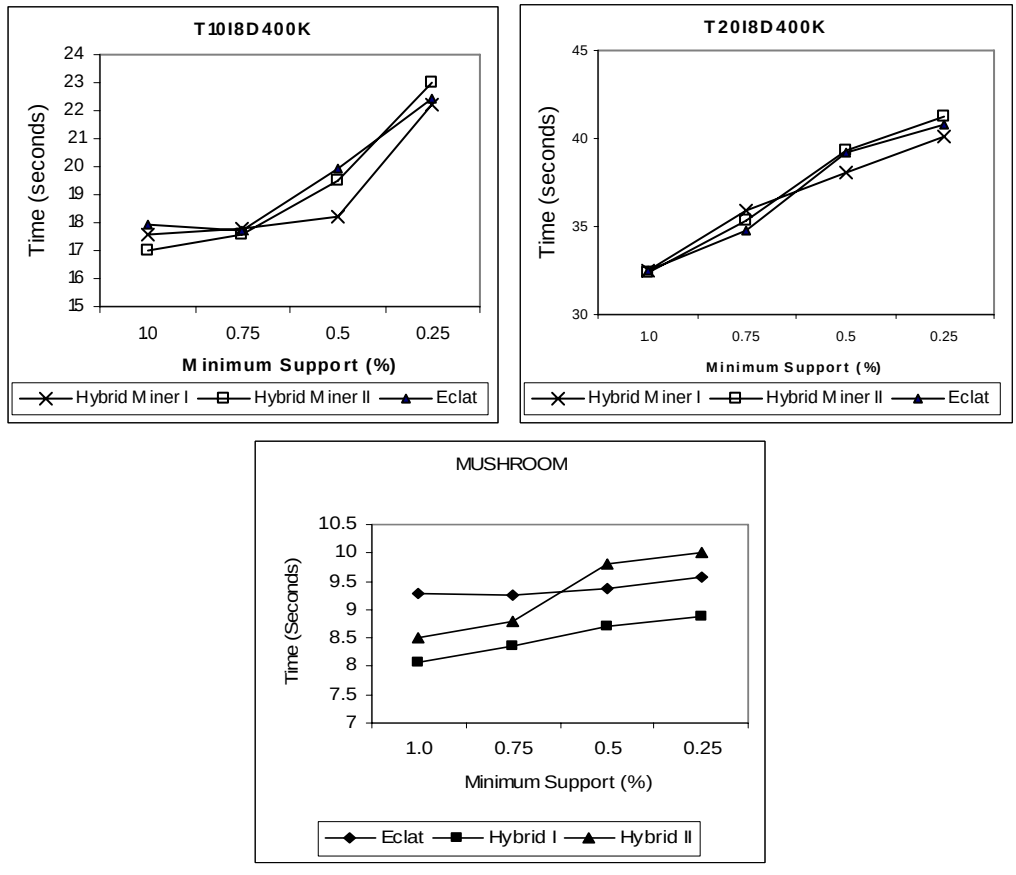


Fig. 6 : Comparative Performance of Hybrid Strategies with Eclat

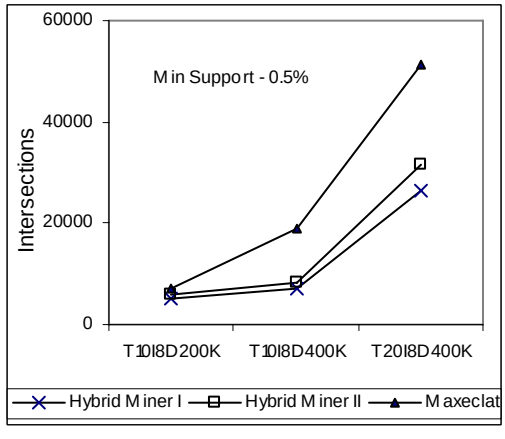


Fig. 7. Tid List Intersections

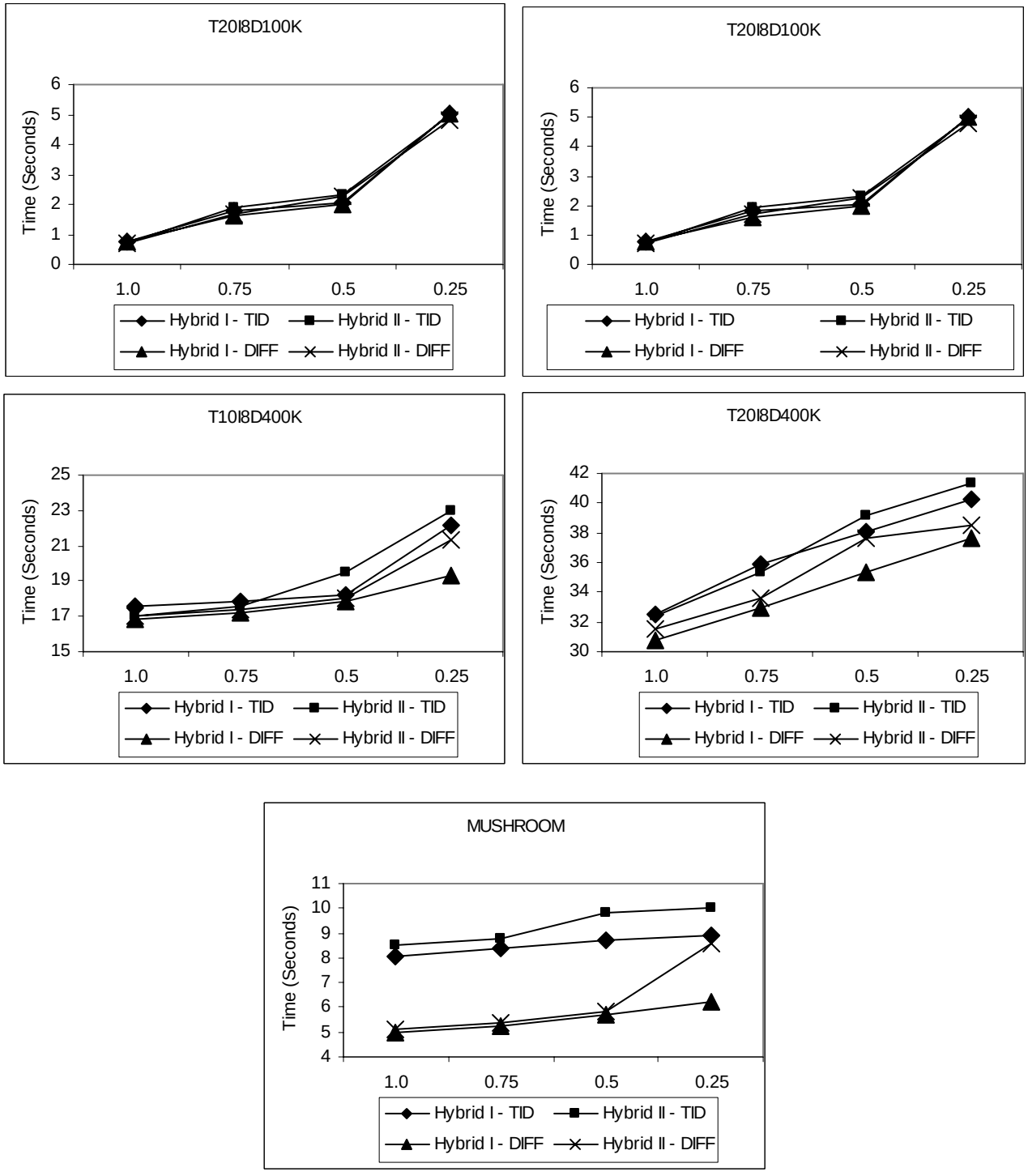


Fig. 8 : Comparative Performance of Hybrid Strategies on TIDsets and Diffsets

ACKNOWLEDGEMENT

We would like to thank Prof. .M.J. Zaki for providing the link to the source code of Maxeclat and Eclat.

REFERENCES

[1] A.Ceglar and J.F.Roddick. "Association Mining", ACM Computing Surveys,vol.38,No.2,July,2006.
[2] R. Agrawal, T. Imielinski and A. Swami, "Mining Association Rules Between Sets of Items in Large Databases", ACM SIGMOD Conf. Management of Data, May, 1993.
[3] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen and A. Inkeri Verkamo, "Fast Discovery of Association Rules", Advances in Knowledge Discovery and Data Mining, U. Fayyad and *et al*., eds. Pp. 307-328, Menlo Park, Calif. : AAAI Press, 1996.
[4] S. Brin, R. Motwani, J. Ullman and S. Tsur, "Dynamic Itemset Counting and Implication Rules for Market Basket Data", ACM SIGMOD Conf. Management of Data, June 1997.
[5] B.A. Davey and H.A. Priestley, Introduction to Lattices and Order, Cambridge Univ. Press, 1990
[6] Han, J., M. Kamber, "Data Mining Concepts and Techniques", Morgan Kaufmann Publishers, 2001.
[7] Hand, D., H. Mannila and P. Smyth, Principles of Data Mining, Eastern Economy Edition, PHI, 2004.
[8] D.I. Lin and Z.M. Kedem, "Pincer Search " A New Algorithm for Discovering the Maximum Frequency Set", Sixth Int'l Conf. Extending Database Technology, Mar. 1998.
[9] J.L.Lin and M.H. Dunham, "Mining Association Rules: Anti-skew Algorithms", 14th Int'l Conf. Data Eng., Feb., 1998.
[10] C. Lucchese, S. Orlando, P. Palmerini, R. Perego and F.Silvestri, "kDCI: A Multistrategy algorithm for Mining Frequent Sets", IEEE ICDM workshop on frequent Itemset Mining Implementation,2003.
[11] A. Mueller, "Fast Sequential and Parallel Algorithms for Association Rule Mining: A Comparison", Technical Report CS-TR-3515, Univ. of Maryland, College Park, Aug. 1995.
[12] Pujari. A.K. Data Mining Techniques, University Press, 2004.
[13] S. Sarawagi, S. Thomas, and R. Agrawal, "Integrating Association Rule Mining with Databases: Alternatives and Implications", ACM SIGMOD Int'l Conf. Management of Data, June 1998.
[14] Savasere, E. Omiecinski and S. Navathe, "An Efficient Algorithm for Mining Association Rules in Large Databases", Proc. 21st Very Large DataBases Conf., 1995.
[15] P.Shenoy ,et al."Turbo Charging Vertical Mining of Large databases",Intl.Conf. Management of Data,May, 2000.
[16] T.Uno, T. Asai, Y. Uchida and H. Arimura, "LCM: An Efficient Algorithm for Enumerating Closed Frequent Itemsets", Workshop on Frequent Itemset Mining Implementation (FIMI 03).
[17] T. Uno, M. Kiyori and H. Arimura ,"LCM Ver 2: Efficient Mining Algorithms for Frequent Closed Maximal Itemsets", Workshop on Frequent Itemset Mining Implementation (FIMI 04).
[18] T. Uno, M. Kiyori and H. Arimura ,"LCM Ver 3: Collaboration of Array Bitmap and FP tree for Frequent Itemset Mining", Conference of Knowledge Discovery in Data, Proceedings of First International Workshop on Open Sources Data Mining :Frequent Pattern Mining Implementations,pg.77-86,2005
[19] P. Valtchev, R. Missaoui, R. Godin and M. Meridji, "Generating Frequent Itemsets Incrementally: Two Novel Approaches based on Galois Lattice Theory", Computational Intelligence, vol 15,1999.
[20] S.J. Yen and A.L.P. Chen, "An Efficient Approach to Discovering Knowledge from Large Databases", Fourth Int'l Conf. Parallel and Distributed Information Systems, Dec. 1996.
[21] M.J. Zaki, S. Parthasarathy, M. Ogihara and W. Li, "New Algorithms for Fast Discovery of Association Rules", Third Int'l Conf. Knowledge Discovery and Data Mining, Aug. 1997.
[22] M.J. Zaki, "Scalable Algorithms for Association Mining", IEEE Transactions on Knowledge and Data Engineering", vol.12 no.3, pp.372-390,May/June,2000.
[21]M.J.Zaki,K.Gouda,"Fast Vertical Mining Using Diffsets", SIGKDD', August 2003.