Ramp: High Performance Frequent Itemset Mining with Efficient Bit-Vector Projection Technique

Shariq Bashir and Abdul Rauf Baig

FAST-National University of Computer and Emerging Sciences, Islamabad, Pakistan
shariqadel@yahoo.com, rauf.baig@nu.edu.pk

Abstract. Mining frequent itemset using bit-vector representation approach is very efficient for small dense datasets, but highly inefficient for sparse datasets due to lack of any efficient bit-vector projection technique. In this paper we present a novel efficient bit-vector projection technique, for sparse and dense datasets. We also present a new frequent itemset mining algorithm *Ramp* (Real Algorithm for Mining Patterns) using bit-vector representation approach and our bit-vector projection technique. The performance of the *Ramp* is compared with the current best frequent itemset mining algorithms. Different experimental results on sparse datasets show that mining frequent itemset using *Ramp* is faster than the current best algorithms.

1 Introduction

Association rules mining introduced by Agrawal [1], has now become one of main pillar of data mining and knowledge discovery tasks, and it is successfully applied in sequential pattern mining, emerging pattern mining, multidimensional pattern mining, classification, maximal and closed pattern mining. Using the support-confidence framework, the problem of mining the complete association rules from transactional dataset is divided into two parts – (a) finding frequent itemsets, and (b) generating association rules from frequent itemsets. Among them part (a) is considered to be the most time consuming process, requires heaviest frequency counting operation for each itemset.

As indicated in [8], MAFIA a maximal itemset mining algorithm [3] (using bit-vector representation approach) is considered to be most efficient algorithm for small dense dataset mining. The main components of MAFIA are its traversal of search space by depth first search, filtering infrequent items from node's tail by dynamic reordering [3] and frequent items representation using vertical bit-vectors. To check the frequency (support) of itemsets it performs a bitwise-AND (bitwise-$\wedge$) operation on head and tail item bit-vectors regions. Since 32-bit CPU supports 32-bit $\wedge$ per operation, hence each region of item bit-vector is composed of 32-bits (*represents 32 transactions*). Calculating frequency using bit-vectors is efficient when the dataset is dense, but highly inefficient when the items bit-vectors contain more zeros than ones, resulting in many useless counting operations, which usually happens in the case of sparse datasets. To handle the bit-vectors sparseness problem, MAFIA proposed a bit-vector projection technique known as projected bitmap. The main deficiency of pro-

W.K. Ng, M. Kitsuregawa, and J. Li (Eds.): PAKDD 2006, LNAI 3918, pp. 504 – 508, 2006.

jection using projected bitmap technique is that, it requires a high processing cost (time) for its creation. Due to this reason, MAFIA used adaptive compression [4], since projection is done only when saving from the compressed bitmaps outweigh the cost of projection. However, with adaptive compression, projection cannot be possible on all nodes of search space.

In this paper we present a novel bit-vector projection technique, we call it *Projected-Bit-Regions* (PBR) bit-vector projection technique and its implementation *Ramp (itemset mining algorithm)*. The main advantages of projection using PBR are that – (a) it consumes a very small processing cost and memory space for projection, and (b) it can be easily applied on all nodes of search space without requiring any adaptive approach. In section 2.2 to 2.4 we present some efficient implementation techniques for *Ramp*, which we experienced in our implementation. Our different experimental results on sparse dataset suggest that mining frequent itemset using *Ramp* is faster than the current best algorithms which marked good scores on FIMI03 and FIMI04 [7]: fpgrowth-zhu [6], AFOPT [8], PatriciaMine [10], AIM [5], MAFIA [3], fpgrowth-borgelt, Eclat-borgelt [2], Apriori-borgelt [2]. This shows the effectiveness of our PBR bit-vector projection technique.

2 Bit-Vector Projection with PBR (Projected-Bit-Regions)

For efficient projection of bit-vectors, the goal of projection should be such as, to bitwise-$\wedge$ only those regions of head bit-vector $\langle$bitmap(head)$\rangle$ with tail item X bit-vector $\langle$bitmap(X)$\rangle$ which contains a value greater than zero and skip all others. Obviously for doing this, our counting procedure must be so powerful and have some information which guides it, that which regions are important and which ones it can skip. To achieve this goal, we propose a novel bit-vector projection technique PBR (Projected-Bit-Regions). With projection using PBR, each node Y of search space contains an array of valid region indexes PBR$\langle$Y$\rangle$ which guide the frequency counting procedure to traverse only those regions which contain an index in array and skip all other. Figure 1 show the code of itemset frequency calculation using PBR technique. As clear from Figure 1, line 2 first retrieves a valid region index ℓ in $\langle$bitmap (head)$\rangle$ and line 3 apply a bitwise-$\wedge$ on $\langle$bitmap (head)$\rangle$ with $\langle$bitmap (X)$\rangle$ on region ℓ.

One main advantage of bit-vector projection using PBR is that, it consumes a very small processing cost for its creation, thereby can be easily applied on all nodes of search space. At any node, projection of child nodes can be created either at the time of frequency calculation if pure depth first search is used, or at the time of creating head bit-vector if dynamic reordering is used. The strategy of creating $PBR_{\langle X \rangle}$ at node n for each tail item X is that, when the PBR of $\langle bitmap(n) \rangle$ are bitwise-$\wedge$ with $\langle bitmap(X) \rangle$ a simple check is perform on each bitwise-$\wedge$ result. If the value of result is greater than zero, then an index is allocated in $PBR_{\langle n.head \cup X \rangle}$. The set of all indexes which contain a value greater than zero makes the projection of $\{head \cup X\}$ node.

2.1 Ramp: Itemset Mining Algorithm

The basic strategy of *Ramp* for mining frequent itemset is that, it traverses search space in depth first order. At any node n, infrequent items from tail are removed by

dynamic reordering and new node m for every tail item X in tail n, is generated such as $m.head = n \cup X$ and $m.tail = n.tail - X$. Items in $m.tail$ are reordered by increasing support which keeps the search space as small as possible. For frequency counting, item X bit-vector is bitwise-$\wedge$ with $n.head$ bit-vector on $PBR_{\langle n \rangle}$. The pseudo code of *Ramp* is described in Figure 1.

Ramp (Node n)
(1) for each item X in n.tail
(2) for each region index ℓ in $PBR_{\langle n \rangle}$
(3) AND-result = bit-vector[ℓ] ∧ head-bit-vector of n [ℓ]
(4) support[X] = support[X] + number of ones(AND-result)
(5) remove infrequent items from n.tail, reorder them by increasing support
(6) for each item X in n.tail
(7) m.head = n.head ∪X
(8) m.tail = n.tail – X
(9) for each region index ℓ in $PBR_{\langle n \rangle}$
(10) AND-result = bit-vector[ℓ] ∧ head-bit-vector[ℓ]
(11) if AND-result > 0
(12) insert ℓ in $PBR_{\langle m \rangle}$
(13) head bit-vector of m [ℓ] = AND-result
(14) Ramp (m)

Fig. 1. Pseudo code of *Ramp* for mining all frequent itemset

2.2 Increasing Projected Bit-Regions Density

The bit-vector projection technique described in section 2 does not provide any compaction or compression mechanism to increase the density in bit-vector regions. As a result, on the sparse dataset only one or two bits are set in each bit-vector region, which not only increase the projection length but also it is not possible to achieve true 32bit CPU performance. To increase the density in bit-vector regions the *Ramp* starts with an array-list [9]. Next at root node, a bit-vector representation for each frequent item is created which provide a sufficient compression and compaction in bit-vectors regions. Sufficient improvements are obtained in *Ramp* by using this approach.

2.3 2-Itemset Pair

There are two methods to check whether current itemset is frequent or infrequent – (a) to directly compute its frequency from its PBR (b) by 2-Itemset pair. If any 2-Itemset pair of any itemset is found infrequent, then by following Apriori [1] property itemset is consider to be as infrequent. In AIM [5] almost the same approach was used with the name *efficient initialization*. However AIM used this approach only for those itemsets which contain a length equal to two. In *Ramp* we extend the basic approach and apply 2-Itemset pair approach also on those itemsets which contain a length more than two. We know any itemset which contains a length more than two, is the superset of its entire 2-Itemset pairs. Before counting its frequency from *TDB*, *Ramp* checks its

2-Itemset pairs. If any pair is found infrequent then that itemset is automatically considered to be infrequent

2.4 Writing Frequent Itemsets to Output File

When the dataset is dense and contains millions of frequent itemsets on low support threshold, almost 90% of overall mining time is spent on writing frequent itemsets to output file. We have noted hat some of previous implementations e.g. AFOPT [8], PatriciaMine [10], fpgrowth-zhu [6] write output itemsets one by one, which increases the context switch and disk rotation times and degrades their algorithm performance. A better approach which we use in *Ramp* is to write itemsets to output file only when a sufficient number of itemsets are mined in memory. In *Ramp* we find that, writing itemsets using this approach sufficiently decreases the processing time of algorithm.

3 Computation Experiments

The implementation of Ramp-all is coded in C language, and the experiments are done on Pentium4 3.2 GHz CPU with 512MB memory. The performance measure is the execution time of the algorithms datasets with different support thresholds on two benchmark datasets (available at http://fimi.cs.helsinki.fi/data/). Figures 2 and 3 show the performance curves of all algorithms. As we can see from Figures, the *Ramp-all* outperforms the other algorithms on almost all support level thresholds, and gives global best performance. The performance improvements of *Ramp-all* over other algorithms are significant at low support thresholds.

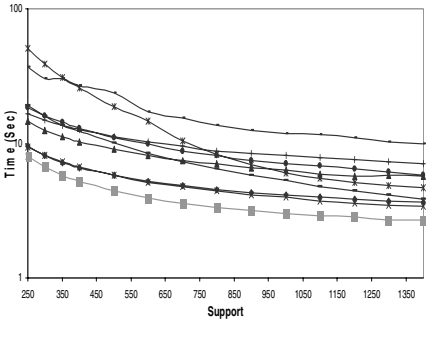


Fig. 2. BMS-POS

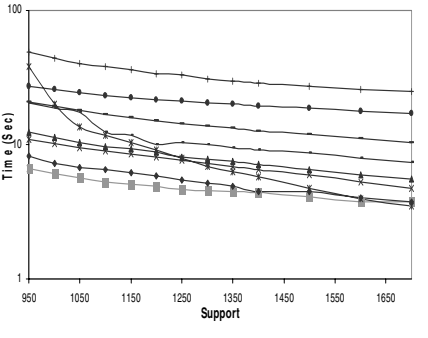
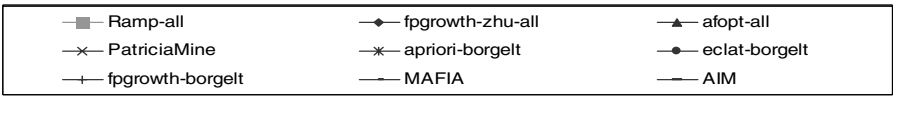


Fig. 3. T40I10D100K



4 Conclusion

Mining frequent itemset using bit-vector representation approach is very efficient for dense datasets, but highly inefficient for sparse datasets due to lack of any efficient

bit-vector projection technique. In this paper we present a novel efficient bit-vector projection technique, which is better than the previous projected bitmap projection technique. The main advantages of our bit-vector projection technique are that, it does not require any rebuilding threshold or does not depend on any adaptive approach for projection, and can be easily applicable on all nodes of search space. We also present a new frequent itemset mining algorithm *Ramp* using our bit-vector projection technique. Different experiments on benchmark datasets show that *Ramp* is faster than the current best frequent itemset algorithms, which show the effectiveness of our bit-vector projection technique.

References

1. R. Agrawal, R. Srikant. Fast algorithms for mining association rules. In *VLDB'94*, Santiago, Chile, 1994.
2. C. Borgelt. Efficient Implementation of Eclat and Apriori. In *IEEE ICDM'03 Workshop FIMI'03*, Melbourne, Florida, USA, 2003.
3. D. Burdick, M. Calimlim, J. Gehrke. MAFIA: A Maximal Frequent Itemset Algorithm for Transactional Databases. In *ICDE'01*, Heidelberg, Germany, 2001.
4. D. Burdick, M, Calimlim, J. Flannick, J. Gehrke, T. Yiu. MAFIA: A Performance Study of Mining Maximal Frequent Itemsets. In *IEEE ICDM'03 Workshop FIMI'03*, Melbourne, Florida, USA, 2003.
5. A. Fiat, S. Shporer. AIM: Another Itemset Miner. In *IEEE ICDM'03 Workshop FIMI'03*, Melbourne, Florida, USA, 2003.
6. G. Grahne, J. Zhu. Efficiently Using Prefix-trees in Mining Frequent Itemsets. In *IEEE ICDM'03 Workshop FIMI'03*, Melbourne, Florida, USA, 2003.
7. Proc. IEEE ICDM Workshop Frequent Itemset Mining Implementations, B. Goethals, M.J. Zaki, eds. CEUR Workshop Proc., vol. 80, Nov. 2003, http://CEUR-WS.org/Vol-90.
8. G. Liu, H. Lu, J.X. Yu, W. Wei, X. Xiao. AFOPT: An Efficient Implementation of Pattern Growth Approach. In *IEEE ICDM '03 Workshop FIMI'03*, Melbourne, Florida, USA, 2003.
9. J. Pei, J. Han, H. Lu, S. Nishio, S. Tang, D. Yang. H-Mine: Hyper-structure mining of frequent patterns in large databases. In *ICDM'01*, San Jose, California, USA, 2001.
10. A. Pietracaprina, D. Zandolin. Mining Frequent Itemsets using Patricia Tries. In *IEEE ICDM'03 Workshop FIMI'03*, Melbourne, Florida, USA, 2003.