PROJECTION BASED SAMPLING FOR MORE EFFICIENT HIGH UTILITY ITEMSET MINING

Alva Erwin, Raj P. Gopalan and N.R. Achuthan*
Department of Computing, Curtin University of Technology
**Department of Mathematics and Statistics, Curtin University of Technology*
Kent St. Bentley, Western Australia

ABSTRACT

High Utility Itemset Mining is a generalization of Frequent Itemset Mining, where not only the absence or the presence of items, but also the utility of items in the form of quantity and profit are significant. Mining High Utility patterns is a difficult problem especially from a large database due to the combinatorial explosion of patterns to be considered and the inapplicability of the downward closure property for pruning. Sampling can reduce the size of the dataset to be mined, but its usefulness depends on the accuracy of the result and the level of accuracy required for a given purpose. In this paper we propose a projection based sampling algorithm to mine High Utility Itemsets that improves the accuracy of mining compared to simple random sampling. Experiments have been performed on real and synthetic datasets to show the effectiveness of the proposed algorithm.

KEYWORDS

High Utility Itemset Mining, Sampling, Frequent Itemset Mining

1. INTRODUCTION

Frequent itemset mining [Agrawal et al. 1993; Agrawal and Srikant 1994] is one of the most intensively researched problems in data mining. Mining high utility itemsets is an extension of the frequent itemset mining problem, but it poses a greater challenge because the *downward closure property* that is used to reduce the search space of potential patterns in *Frequent Itemset* mining does not hold good for high utility patterns. A framework for high utility itemset mining was originally proposed by Yao et al [Yao et al. 2004; Yao and Hamilton 2006]. They described a mining method and pruning strategies based on the mathematical properties of utility constraints. Their algorithm named *Umining* and a heuristic based algorithm *Umining_H* adapted the level wise mining strategy of the Apriori algorithm to discover high utility itemsets. Recent research has focused on efficient high utility mining using intermediate anti-monotone measures for pruning the search space. Liu et al. [Liu et al. 2005] proposed a two phase algorithm to mine high utility itemsets, where they use a transaction weighted utility (TWU) measure in the first phase to find supersets of high utility itemsets, followed by a rescan of the database to determine the actual high utility itemsets among them.

Similar to Frequent Itemset Mining, finding High Utility Itemsets is a difficult problem due to the combinatorial explosion of potential itemsets especially in large datasets. Sampling can reduce the size of the dataset and thereby give approximate results with faster time and less memory requirements. In many applications such as social networking websites and e-commerce recommendation systems, obtaining approximate results quickly is preferred over getting exact results that take a longer time to compute.

The rest of this paper is organized as follows: Section 2 defines the problem and briefly reviews the related work. Section 3 describes the proposed algorithm. Section 4 reports the experimental results and Section 5 contains the conclusion and some directions for future research.

2. PROBLEM DEFINITION

In this section, we give the basic notations and the definitions of terms to describe high utility itemset (HUI) mining, based on [Yao et al. 2004; Liu et al. 2005; Yao et al. 2006] and describe the existing work related to our research.

2.1 High Utility Itemset Mining

Let $I=\{i_1,i_2,...,i_m\}$ be a set of items and $DB=\{T_1,T_2,...,T_n\}$ be a transaction database where the items of each transaction T_q is a subset of I. Unlike conventional Frequent Itemset mining where only the presence or absence of an item is recorded, a transaction database for High Utility Itemset mining also records the quantity of each item in a transaction, as well as the external utility (e.g. profit per unit) of the item. The quantity of an item i_p in a transaction T_q is denoted by $o(i_p,T_q)$. The external utility $s(i_p)$ is the value of a unit of item i_p in the utility table. The utility of item i_p in transaction T_q, denoted by $u(i_p,T_q)$ is defined as $o(i_p,T_q) \times s(i_p)$. A set X is called an itemset if X is a subset of I. The utility of X in transaction T_q, denoted by $u(X,T_q)$ is defined as :

$$u(X,T_q) = \sum_{i_p \in X} u(i_p,T_q) \tag{1}$$

The utility of itemset X in the database, denoted by $u(X)$ is defined as:

$$u(X) = \sum_{T_q \in D \wedge X \subseteq T_q} u(X,T_q) \tag{2}$$

TID	A	B	C	D	E	F	Transaction Utility ($)
t_1	2	0	1	1	0	0	80
t_2	2	1	1	0	0	0	195
t_3	0	0	1	1	10	0	110
t_4	0	1	0	0	15	0	225
t_5	1	0	1	0	0	1	37
t_6	2	0	0	1	10	0	105
t_7	2	0	0	0	8	1	62
t_8	1	1	0	1	2	0	205
t_9	1	0	0	1	10	0	95
t_{10}	1	1	0	0	5	0	185
total	12	4	4	5	60	2	1299

(a) Transaction database

Symbol	Item Name	Profit ($)
A	T oner	10
B	Colour Laser Printer	150
C	Bubble Jet Printer	25
D	Digital Camera	35
E	Glossy Photo Paper	5
F	USB drive	2

(b) Utility table

Figure 1. An example transaction database and utility table

The task of high utility itemset mining is to find all itemsets that have utility above a user-specified *minUtil*. Since utility is not *anti-monotone,* Liu et al. [Liu et al. 2005] proposed the concepts of Transaction Utility (TU) and Transaction Weighted Utility (TWU) to prune the search space of high utility itemset. Transaction Utility of a transaction, denoted $tu(T_q)$ is the sum of the utilities of all items in T_q:

$$tu(T_q) = \sum_{i_p \in T_q} u(i_p, T_q) \tag{3}$$

Transaction Weighted Utility of an itemset X, denoted as $twu(X)$ is the sum of the transaction utilities of all the transactions containing X:

$$twu(X) = \sum_{T_q \in D \wedge X \subseteq T_q} tu(T_q) \tag{4}$$

As shown in [Liu et al. 2005], any superset of a low TWU itemset is also a low TWU itemset. Thus we can use TWU to prune the supersets of low TWU itemsets. However, since TWU is an over-estimation of the real utility itemset value, further pruning of high TWU itemsets will be required for mining high utility itemsets. Suppose we have a small transaction database of an electronic retailer as shown in Figure 1a, and Figure 1b with the profit (external utility) for each item. The values in each row of Figure 1a shows the quantity of each item bought in a particular transaction. The last column shows the transaction utility of each transaction with the total transaction utility of the database in the last row. In transaction t_1, two A (printer ink), one C (bubble jet printer), and one D (digital camera) were bought, yielding a transaction utility of \$80. The utility of item A, $u(A,t_1)$ is \$20 and the utility of item A in the whole database, $u(A)$ = \$120. Itemset CD occurs 2 times, in transactions t_1 and t_3.

Further, $u(CD,t_1)$ = \$60, $u(CD,t_3)$ = \$60 and $u(CD)$ = \$120 and $twu(CD)$=\$190. The TWU of item B (colour laser printer) is the sum of the transaction utilities of (t_2, t_4, t_8, t_{10}) = \$810 and the TWU of itemset AC is the sum of transaction utilities of (t_1, t_2, t_5) = \$ 312. Suppose we are interested in itemsets with *minUtil* as 10% of the total database utility (=\$129.9). Then itemset AC has low utility, whereas itemset AB is of high utility. We define any singleton item having TWU larger than or equal to *minUtil* as 1-hTWU. In the above example, item B is a 1-hTWU, because $twu(B)$ is \$810.

2.2 Related Work

Several researchers have studied sampling based methods for frequent itemset mining, but very little work has been reported on the use of sampling based approaches for high utility mining. Zaki et al. studied the simple random sampling without replacement (SRSWR) to draw samples from the original database [Zaki et al. 1997]. Their paper analyzes Chernoff Bounds and shows how to determine a sample size based on the bound frequency of itemset, independent from the original dataset. Li and Gopalan studied the Central Limit Theory to determine the sample size for Association Rule Mining [Li and Gopalan 2004]. Toivonen proposed a two phase approach, where in the first phase, the frequent itemset is mined from sampled database [Toivonen 1996]. To reduce the probability of missing frequent itemsets (false positive error), the minimum support threshold was lowered. In the second phase, the frequent itemsets discovered in the first phase are checked for their true frequency in the original dataset. [Chen et al. 2002] propose a novel two phase sampling method called Finding Associations from Sampled Transaction (FAST). In the first phase of their method, a large sample is taken from the original dataset, and in the second phase, the final sample is obtained by trimming the outlier transactions until the sample reaches a predefined size. [Shengnan et al. 2005] proposed a framework for parallel data mining of frequent itemsets and sequential patterns. They introduce selective sampling to address the problem of load balancing in the parallel data mining environment. To the best of our knowledge, this is the first attempt to use parallel projected sampling to deal with the High Utility Itemset Mining problem.

3. PROPOSED ALGORITHM

The parallel projected sampling method we propose in this paper is based on the idea of database projections reported in [Pei 2002]. In this section, we describe our proposed algorithm for mining High Utility Itemsets by parallel projected sampling from a transaction database.

3.1 Algorithm Description

Our sampling method is based on parallel projection of the transaction database based on the presence each 1-hTWU itemset as described below. The projection scheme is similar to that proposed in [Pei 2002] and [Shengnan et al. 2005] for frequent itemsets. The procedure is divided into four steps, as illustrated in Figure 2.

1) Identify the 1-hTWU itemsets in the transaction database, based on the user supplied *minUtil* value. The algorithm to compute 1-hTWU itemsets is given in Figure 3a.

2) Subdivide the transaction database into n projections $p_1..p_n$, based on the occurrence of 1-hTWU items. Note that each projection may have a different size, due to the number of occurrence of the 1-hTWU item in the projection. The algorithm to compute the parallel projection, based on the 1-hTWU itemset is given in Figure 3b.
3) Use simple random sampling on each projection to get a set of smaller projections.
4) Perform the HUI mining on each sampled projection, and combine it into the final result.

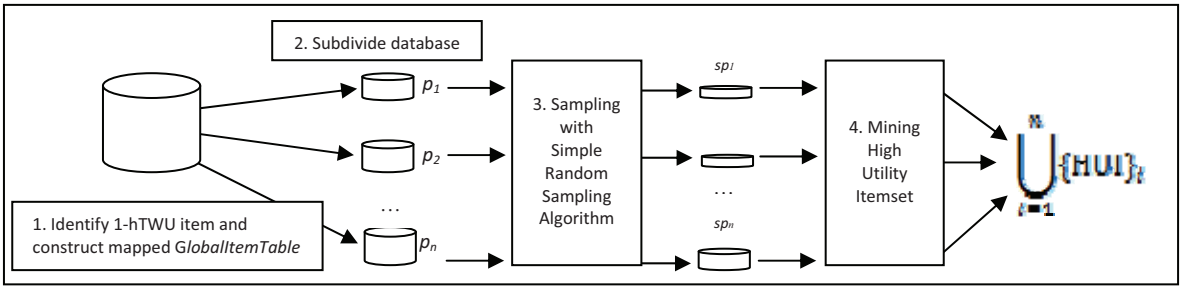

Figure 2. Sampling from projected database

```
Algorithm : Construct 1-hTWU itemset
Input : Transaction DB, external utility,
Parameter : minimum utility
Output : Array 1-hTWU items GlobalItemTable

1: begin
2:   initialize GlobalItemTable
3:   for each transaction t ∈ DB
3:     for each item i ∈ t
4:       if i ∈ GlobalItemTable
5:         Increment TWU of i
6:       else
7:         insert i into GlobalItemTable
           with TWU of i
8:       end if
9:     end for
10:  end for
11:  sort GlobalItemTable in TWU descending
     order
12:  if TWU(i) < minimum utility
13:     prune item i from GlobalItemTable
14:  else
15:     assign index to item i
16:  end if
```

(a) Finding 1-hTWU itemset

```
Algorithm : Parallel Projection
Input : Array 1-hTWU items GlobalItemTable
        Transaction DB
Output : n Projected DB

1: begin
2:   for each transaction t ∈ DB do
3:     map each item in transaction to
       its index number in GlobalItemTable
4:     sort transaction based on index
5:     for each index number in trans do
6:       write suffix of the transaction
         starting from current index to
         projection_index
7:     end for
8:  end for
```

(b) Parallel Projection of Transaction Database

Figure 3. 1-hTWU and parallel projection Algorithm

3.2 Detailed Example

Using the sample database in Figure 1 and *minUtil* as 10% of the original utility of the database, we illustrate our method by the following example.

Step 1. Identify 1-hTWU items

The first step is to find 1-hTWU items based on the sample transaction database and utility table. Each transaction is a tuple containing occurrences of several items in the form <item (quantity)>. The first transaction t_1 is read as the following record: t_1 A(2), C(1), D(1). This transaction has transaction utility of 80 as shown in Figure 1a. The item A, C and D will then be inserted into the *GlobalItemTable*, with TWU of 80 for each item (see Figure 4a). After processing the second transaction t_2 A(2), B(1), C(1) with the transaction utility of 195, we will modify the TWU of item A (previously 80) to 275, TWU of item C (previously 80) to 275 and insert a new item B into *GlobalItemTable* with 195 as the current TWU of item B. Since item D does not occur in the transaction t_2, item D's TWU remains as 80 (see Figure 4b). When we finish reading all the transactions, *GlobalItemTable* will be as shown in Figure 4c. Next, *GlobalItemTable* is sorted on TWU, item

F is pruned as it falls below threshold and new index values assigned to the items. The final *GlobalItemTable* is shown in Figure 4d.

Item	A	B		C	D	E		F
TWU	80	0	80		80		0	0

(a) *GlobalItemTable* after processing t_1

Item	A		B		C	D	E	F
TWU	275	19	5	27	5	80	0	0

(b) *GlobalItemTable* after processing t_2

Item	A		B		C	D	E	F
TWU	964	8	10	4	22	5	95	9	87	37

(c) *GlobalItemTable* after processing t_{10}

Index	1	2	3	4	5		
Item	E	A	B			D	C
TWU	987	964	8	10	5	95	4	22

(d) Final *GlobalItemTable*

Figure 4. *GlobalItemTable* construction in progress

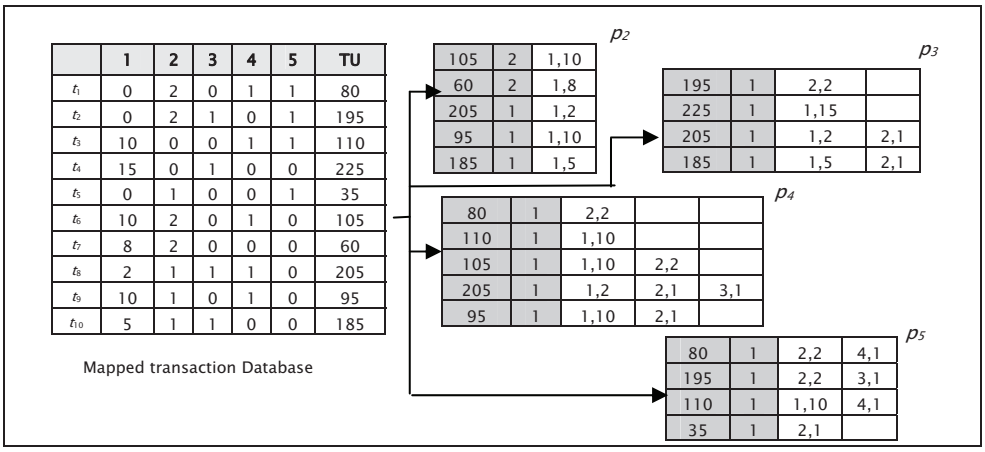

Figure 5. 1-hTWU and parallel projection Algorithm

Step 2. Subdivide the transaction database

Using the *GlobalItemTable* of Figure 4d, the original database is transformed into the mapped transaction database. Concurrently the parallel projections are constructed. Figure 5 illustrates the process for the database of Figure 1 using the corresponding *GlobalItemTable*. For item index $i \geq 2$, every transaction t with positive quantities of items up to item i is written into the projection p_i with the corresponding quantities and TWU. So there are five entries in p_2 from transactions t_6, t_7, t_8, t_9 and t_{10}. Similarly, in p_3, we have entries for item index 1 and/or 2 that occur before index 3 in the transactions t_2, t_4, t_8 and t_{10}. The third projection (p_4) is from transactions t_1, t_3, t_6 t_8, and t_9, and the last projection is from transactions t_1, t_2, t_3, and t_5.

Step 3. Sampling the projection data

In this step, we apply the Simple Random Sampling to the projected database. We draw sample from projections p_1.. p_5 to get sp_1.. sp_5. For the sake of simplicity, we will use a sample size of 85% of the original partition size in this example. For the larger datasets used in our experimental evaluation, the sample sizes will be much lower. Suppose after sampling, we got sample projection sp_5 containing the set of mapped transactions as follows : {{2 ,4,5},{2,3,5},{1,4,5}}, thus we miss the transaction containing the mapped transaction items {2,5}. We continue to the next step of mining from the sampled projection.

Step 4. High Utility Itemset Mining

In this step, we perform the High Utility Itemset Mining on each individual projected sample sp_i by using the CTU-PROL algorithm [Erwin et al. 2008]. The results from mining all the projections are combined to get the final approximation of High Utility Itemsets. We lower the value of *minUtil* according to the percentage of sample size relative to the projected database.

4. EXPERIMENTAL EVALUATION

In this Section, the implementation of our algorithm is empirically tested. The program is written in C++ and compiled using g++ version 4.1.0. All the testing was performed on a Pentium Duo Core, 3.2 GB RAM, with

Linux Fedora Core 11 operating system. For the experimental evaluation of our method, we use two synthetic datasets and modified versions of real-world datasets BMSPOS and Mushroom that are available from FIMI (http://fimi.cs.helsinki.fi/). The characteristics of our datasets are shown in Table 1.

Table 1. Characteristics of Datasets

Dataset	# of Transactions	# of Item	Size of file (MB)
T10I100D50K 50,	000	100	4.5
T5I100D100K 10	0,000	100	5
modBMSPOS 51	5,597	1657	30
modMushroom 81	24	119	1.5

Table 2. Experiment results for Synthetic datasets

		10%	20%	30%	40%	50%	60%
T10I100D50K	minUtil=0.1						
SRS	Time To Mine (sec.)	7.31	14.11	21.65	29.41	36.79	44.57
	Precision	0.6613	0.8361	0.8695	0.9221	0.9282	0.9494
	Recall	0.8356	0.8934	0.8992	0.9309	0.9329	0.9485
PL	Time To Mine (sec.)	13.28	22.416	32.317	41.821	51.684	61.205
	Precision	0.7882	0.9342	0.9643	0.9811	0.9875	0.9927
	Recall	0.9771	0.9852	0.9902	0.9912	0.9942	0.9945
T10I100D50K	minUtil=0.025						
SRS	Time To Mine (sec.)	15.25	24.75	32.58	40.42	47.06	55.17
	Precision	0.5881	0.5881	0.6550	0.7340	0.7550	0.8055
	Recall	0.5886	0.5886	0.6372	0.7095	0.7357	0.7936
PL	Time To Mine (sec.)	34.836	37.836	49.285	60.97	73.177	85.8
	Precision	0.7286	0.8267	0.8988	0.9269	0.9405	0.9500
	Recall	0.7562	0.8297	0.8747	0.8985	0.9191	0.9358
T5I100D100K	minUtil=0.05						
SRS	Time To Mine (sec.)	2.06	4.14	6.37	8.48	10.62	12.77
	Precision	0.8555	0.9039	0.9080	0.9449	0.9441	0.9653
	Recall	0.8070	0.8723	0.9011	0.9299	0.9344	0.9550
PL	Time To Mine (sec.)	6.63	9.39	11.9	14.57	17.139	19.846
	Precision	0.9488	0.9596	0.9632	0.9662	0.9754	0.9806
	Recall	0.8774	0.9289	0.9437	0.9533	0.9681	0.9742
T5I100D100K	minUtil=0.01						
SRS	Time To Mine (sec.)	6.86	7.59	9.98	15.83	15.83	18.84
	Precision	0.3651	0.3651	0.5790	0.6889	0.7301	0.7892
	Recall	0.4568	0.5451	0.6052	0.6859	0.7251	0.7810
PL	Time To Mine (sec.)	12.75	13.44	16.585	20.36	24.256	28.16
	Precision	0.4249	0.4549	0.7198	0.7786	0.8095	0.8437
	Recall	0.6161	0.6933	0.7423	0.7786	0.8095	0.8437

Table 3. Experiment results for Real datasets

10%			20%	30%	40%	50%	60%
modBMSPOS	minUtil=0.1						
SRS	Time To Mine (sec.)	188.45	182.42	223.133	223.46	224.663	282.42
	Precision	0.6730	0.8920	0.9010	0.9357	0.9460	0.9490
	Recall	0.8020	0.8850	0.9480	0.9420	0.9290	0.9460
PL	Time To Mine (sec.)	260.45	264.61	266.98	260.54	281.7	302.867
	Precision	0.8265	0.8933	0.9138	0.9320	0.9459	0.9525
	Recall	0.8742	0.9128	0.9362	0.9447	0.9588	0.9705
modMushroom	minUtil=10.0						
SRS	Time To Mine (sec.)	2.36	2.49	2.736	2.761	2.889	3.08
	Precision	0.2234	0.8626	0.868127	0.8227	0.90558	0.8601796
	Recall	0.4282	0.8661	0.997858	0.6414	0.84762	0.7694162
PL	Time To Mine (sec.)	2.828	2.98	3.15	3.279	3.39	3.43
	Precision	0.8389	0.8283	0.876378	0.8839	0.90237	0.9538343
	Recall	0.6636	0.6652	0.979646	0.9871	0.81682	0.9959829

Since BMSPOS and Mushroom originally consists of only the presence of items in transactions, we generated a utility table (profit per unit) for each item based on lognormal distribution with the utility values ranging from 0.01 to 10. The quantities of items were generated randomly in the range of 1 to 10. We call these datasets modBMSPOS and modMushroom.

To evaluate the quality of our Sampling, we use Precision and Recall as metrics. Suppose that TP is the number of high utility itemsets (HUI) that exist both in the original dataset and the sample, FP is the number of HUI existing in the sample, but not in the original dataset and FN is the number of HUI existing in the original dataset but not in the sample, then Precision P=TP/(TP+FP) and Recall R=TP/(TP+FN). The results of our experiments are shown in Tables 3-4 and Figure 6.

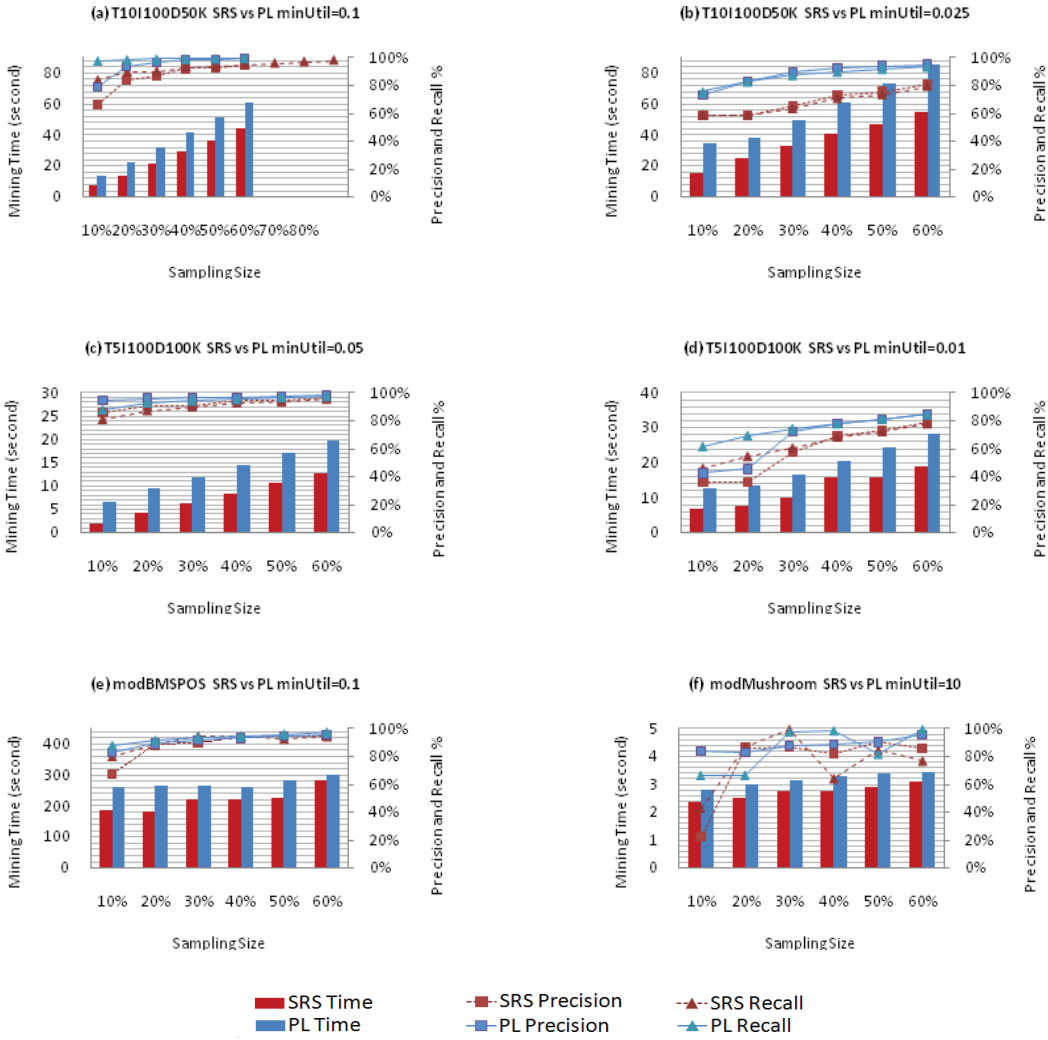

Figure 6. Experiment results for synthetic and real datasets

We abbreviate the Parallel Projection Sampling based method as PL and Simple Random Sampling as SRS in Tables 2- 3 and Figure 6. With the T10I100D50K dataset containing 50,000 transactions, we performed the experiments with two *minUtil* values of 0.1 and 0.025. The results are shown in Table 2, Figure 6a, and Figure 6b. The next synthetic dataset is of 100,000 transactions, with slightly shorter average length of each transaction. The results for this dataset can be seen in Table 2, Figure 6c , and Figure 6d. For the real dataset, we used modBMSPOS and modMushroom mentioned above, which are already embedded with quantity and profit values. The results for these datasets can be seen in Table 3, Figure 6e, and Figure 6f. The numerical results are given in Table 2 and Table 3 for ease of reference to specific values.It can be seen from the results in Tables 2 and 3 that our method in comparison to the results of SRS improved the precision (recall) by around 6 – 18% (3 – 15%), while mining synthetic data sets with sample sizes ranging from 10% to 50%. However, a small additional mining time (10 to 20 seconds) is required for our method compared to SRS. As we can see in Table 2 and Figures 6a-6d, applying our algorithm with a low *minUtil* value (see Figure 6b and Figure 6d) gives better results of precision and recall compared to mining with a high *minUtil* value (see Figure 6a and Figure 6c) on the same dataset.

For the real datasets, the improvement in precision and recall is more significant on smaller sample sizes. In the modBMSPOS dataset, with *minUtil* of 0.1 of total database utility, precision increases from 0.83 for a 10% sample to 0.95 for a 60% sample and the corresponding recall goes from 0.87 to 0.97. For the larger sample sizes, the differences in precision and recall between our method and SRS are smaller. For

modMushroom which is a high dense dataset, the minimum utility is set higher since a very large number of itemsets will be produced at low minimum utility values. For this dense dataset, difference in execution time between SRS and our method is relatively small. As can be expected, the improvement by our method is much smaller while mining a data set with a relatively larger sample size.

5. CONCLUSION

We proposed a new sampling based algorithm to find approximate results from High Utility Itemset mining. We implemented and tested the algorithm using synthetic and real datasets, and compared the results with that from simple random sampling. The accuracy of our method as measured by precision and recall is shown to be significantly better than that of simple random sampling. The limitation of our approach is that when the sample size is too large (greater than about 60%), it takes more time to mine compared to the non-projection based approach as the total size of all partitions is larger than the original dataset.

We plan to extend our sampling method by replacing simple random sampling of projections by weighted sampling based on transaction utility. Our future research will also aim to study the susitability of deterministic sampling proposed in [Bronnimann et al. 2002] and [Akcan et al. 2007].

ACKNOWLEDGMENT

Alva Erwin is supported by Curtin International Research Tuition Scholarship (CIRTS).

REFERENCES

Agrawal, R., T. Imielinski and A. Swami (1993). *Mining association rules between sets of items in large database*. ACM SIGMOD International Conference on Management of Data.

Agrawal, R. and R. Srikant (1994). *Fast algorithms for mining association rules*. 20th VLDB Conference, Santiago Chile.

Akcan, H., A. Astashyn and H. Bronnimann (2007). "Deterministic algorithms for sampling count data." *Data & Knowledge Engineering* 64(2).

Bronnimann, H., B. C. Dash, P. Haas, Y. Qiao and P. Scheuermann (2002). "Efficient data-reduction methods for on-line association rule discovery."

Chen, B., P. Haas and P. Scheuermann (2002). *A new twophase sampling based algorithm for discovering association rules*. SIGKDD '02, Edmonton, Alberta, Canada.

Erwin, A., R. P. Gopalan and N. R. Achuthan (2008). *Efficient mining of high utility itemsets from large datasets*. PAKDD, Osaka, Japan.

Li, Y. and R. P. Gopalan (2004). *Effective sampling for mining association rules*. Australian Conference on Artificial Intelligence, Springer.

Liu, Y., W.-K. Liao and A. Choudhary (2005). *A fast high utility itemsets mining algorithm*. 1st Workshop on Utility-Based Data Mining, Chicago Illinois.

Pei, J. (2002). Pattern-growth methods for frequent pattern mining. *PhD Thesis*, Simon Fraser University.

Shengnan, C., H. Jiawei, H. Jay and P. David (2005). A sampling-based framework for parallel data mining. *Proceedings of the tenth ACM SIGPLAN symposium on Principles and practice of parallel programming*. Chicago, IL, USA, ACM.

Toivonen, H. (1996). *Sampling large databases for association rules*. 22nd VLDB Conference, Bombay.

Yao, H. and H. J. Hamilton (2006). "Mining itemset utilities from transaction databases." *Data & Knowledge Engineering* 59(3): 603 - 626.

Yao, H., H. J. Hamilton and C. J. Buzz (2004). *A foundational approach to mining itemset utilities from databases*. 4th SIAM International Conference on Data Mining, Florida USA.

Yao, H., H. J. Hamilton and L. Geng (2006). *A unified framework for utility based measures for mining itemsets*. ACM SIGKDD 2nd Workshop on Utility-Based Data Mining.

Zaki, M. J., S. Parthasarathy, W. Li and M. Ogihara (1997). Evaluation of sampling for data mining of association rules. Rochester, New York, University of Rochester.