CHAPTER 6

Conclusions

6.1 Summary

In this thesis, we proposed a novel mining task: mining frequent superset, and designed three methods for the new problem. The three methods, Apriori-C, Eclat-C, and data complement technique (DCT) for discovering frequent superset, which are all based on the set complement view of point. We do not find frequent supersets directly, but the complement of frequent supersets, called complement-frequent superset.

Apriori-C is a level-wise, breadth-first search, bottom-up and counting occurrence method that contains the prune-and-join two steps. Eclat-C is a depth-first search and transaction-ID list intersecting method. DCT can take the original frequent itemset mining algorithm as the black box to discover the frequent superset. We performed several experiments to evaluate our three algorithms in the variety of dataset and minimum support. In DCT method, Apriori, Eclat and FP-Growth are taken as the black box to compare with Apriori-C and Eclat-C. We modify the original Apriori algorithm as a Baseline in addition.

The experimental results show that all of these three algorithms are more time efficient than the Baseline method, and especially the Eclat-C is most time efficient in almost every situations. For DCT method, DCT-FPGrowth is the most time efficient algorithm as we

expect. When the number of transactions increases, the execution time of Apriori-C and Eclat-C increase linearly. Since the complement time is raised with the number of transactions growing up, DCT becomes worse. Even DCT-FPGrowth is worse than the Baseline method when the number of transactions reaches to 100k. However, sometimes DCT-FPGrowth can demolish others, when the number of items is larger than the average size of transactions enormously ($|T|/N<0.4$), but the number of transaction is fewer ($|D|<10k$). In this case, the complement time is not significant any more. In addition, Apriori-C beats other algorithms, while the minimum support is large (minsup>90%).

Besides, we also evaluate the performances of the proposed algorithms when both change the two parameters, N and |T|, and we fixed the value of N-|T| as a constant. The results show that when $\frac{|T|}{N}$ increases, the execution gets faster. This is because the occurrence probability of candidate complement-superset becomes larger. However, there is a tradeoff between the occurrence probability of candidates and transaction size. By contraries, when the average size of transactions increases, the performance of the proposed algorithms becomes better. This is also due to the occurrence probability of items.

6.2 Future Work

Future work for frequent superset mining can be divided into two parts. The first is to design other algorithms with new strategies for this problem. Currently, we have tried the breadth-first search, depth-first search, transaction-ID list intersecting, and counting occurrence. In the future, if we just want to look for the frequent superset, and do not care the support value, the DISC [24] strategy can be adopted. The DISC strategy is proposed in

2003, which is to discover frequent subsequence (sequential pattern mining [25]) without counting support.

The second is to discover other frequent super-patterns, such as frequent super-sequence, frequent super-tree, frequent super-graph, and so forth. For example, the fragment assembly problem in bioinformatics [26] can be seen as a frequent super-sequence mining problem, and the ontology merging [27] or schema integration problem can be seen as a frequent super-tree or frequent super-graph problem. In the past, the problems of super-sequence, super-tree, and super-graph are solved by merging two by two, iteratively [28][29][30][31]. For the super-graph problem, we can utilize the concept of complement graph for our algorithm to solve it. However, for super-sequence or super-tree problem, the critical issue is to give a clear definition of complement-sequence or complement-tree.