Mining *N*-most Interesting Multi-level Frequent Itemsets without Support Threshold

Sorapol Chompaisal[1], Komate Amphawan[2], and Athasit Surarerks[1]

[1] ELITE Laboratory, Chulalongkorn University, Bangkok, Thailand
capukampan22@gmail.com, Athasit.S@chula.ac.th
[2] Computational Innovation Laboratory, Burapha Univerisity, Thailand
komate@gmail.com

Abstract. Mining multi-level frequent itemsets from transactional database is one of the most important tasks in data mining community. It aims to discover correlation among items with their hierarchical categories under support-confidence values and thresholds. However, it is well-known that the task of providing an appropriate support threshold to mine the most interesting patterns without prior knowledge in advance is very difficult and it is more reasonable to ask the users to specify the number of desired patterns. Therefore, in this paper, we propose an alternative approach to mine the most interesting multi-level frequent patterns without the setting of support threshold, called *N-most interesting multi-level frequent pattern mining*, where *N* is the number of desired patterns with the highest support values per each category level. To mine such patterns, an efficient adaptive *FP-growth* algorithm, called *NMLFP*, is proposed. Extensive performance studies show that *NMLFP* has high performance and linearly scalable on the number of desired results.

Keywords: Association Rules, *N*-most interesting patterns, Multi-level frequent itemsets.

1 Introduction

Association rule mining (*ARM*) [1, 2] under the support-confidence framework is the task of discovering relationship or correlation among items appearing together in large database. It is applied in a wide range of applications such as marketing, medical diagnostics, web analysis, decision making, etc. The process of *ARM* can be divided two steps: (*i*) discovering frequent itemsets from the given database that meet the support threshold, and (*ii*) generation rules from the frequent itemsets found in the first step that satisfy the confidence threshold.

In general, *ARM* focused on investigation of interesting rules at a single concept level. However, there are some applications which need to discover relations at multiple concept level. For example, besides of the need of finding 75% of customers that purchase milk may also purchase cookies, it could be informative to also illustrate that 65% of people buy almond cookies if they buy chocolate milk. The latter information expresses the lower concept level with more specific and concrete information than that of the former. Therefore, Han et al. [3] introduced an approach to mine multiple-level association rules under user-given concept hierarchy of items and support-confidence thresholds to discover more general/specific knowledge from database in real-world applications.

To effectively and efficiently explore multiple-level concept level of association rules, there are two simple frameworks of defining the support threshold to measure interestingness of itemsets and association rules. Firstly, the framework of setting only one support threshold for all level items (also called *uniform minimum support*) is proposed. Under this approach, it is quite convenience to users to determine a support threshold. However, it may give some uninteresting rules at higher level if the support threshold is small. Then, the setting different minimum support threshold on each level by reducing the thresholds at lower levels of hierarchy, called *reduced minimum support* [3, 4], is introduced to alleviate a drawback of the first framework. This approach cannot only be found interesting rules at different level, but may also have high potential to find nontrivial, informative association rules.

However, it is well-known that the setting of a suitable support threshold without prior knowledge in advance is a difficult task. If the support threshold is too high, then there may be only a small or even no result. In that case, the user may have to guest a smaller threshold and do the mining again. If the threshold is too low, then there may be too many results for the users. Thus, asking the number of desired outputs is considered easier and mining top-*k* frequent patterns/*N*-most interesting patterns [8-11] has become a very popular task.

Thus, in this paper, we aim to alleviate difficulties of setting support threshold and then introduce an alternative approach to mine the most interesting multi-level frequent pattern, called *N-most interesting multi-level frequent itemset mining*. This approach allows the users to specify a simple threshold that is the number of desired result (*N*). Then, a set of *N* most frequent itemsets at each level is discovered. To quickly discover such itemsets, an efficient algorithm namely *NMLFP* is proposed. *NMLFP* is a tree-based algorithm under FP-tree and FP-growth technique. It also applies top-down traversal to quickly cut-down the search space.

The rest of this paper is organized as follows. In Section 2, the concepts related to the multi-level frequent patterns and *N*-most interesting multiple-level frequent patterns are introduced. An efficient method to discover *N*-most interesting multiple-level frequent itemsets are described in Section 3. Section 4 reports the performance analysis under several experiments. Lastly, the conclusion is in Section 5.

2 Problem Definitions

In this section, basic definitions of the concept hierarchy for identifying multi-level frequent patterns and notations for *N*-most interesting multi-level frequent patterns are described.

2.1 Multi-level Frequent Patterns

Let $I= \{i_1, i_2,\ldots,i_m\}$ be a set of products, objects or events, called *items*, where each item is associated with multiple meaning. For example, given the product hierarchy of

Fig. 1, "*foremost chocolate milk*" may be viewed on three different perspectives. At the lowest level (level 1), we can consider "*foremost chocolate milk*" as a specific *product*. At the higher level (level 2), it simply represents the *kind of milk*, meanwhile the highest level (level 3) refers to *milk* which is a kind of food. In this context, the terms of lower and higher level is applied to express the levels of items' meaning.

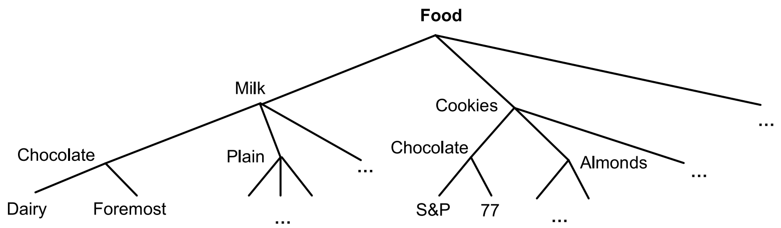


Fig. 1. A concept hierarchy for the relevant data items

A transactional database $TDB = \{t_1, t_2, ..., t_n\}$ is a set of transactions where a transaction t_j is composed of (*i*) a unique transaction identifier (*tid*) and (*ii*) a set of items *Y*, in which each item is encoded into a concise numeric form (also with the sequences of the symbol '*' according to their positions in the hierarchy concept) that can identify not only the given products, but its hierarchical level as well. For example, node of "*Milk*" in Fig.1 would be represented by "1**", the node of "*Chocolate*" after "*Milk*" by "11*" and the last level node of "*Foremost*" by "112", respectively. Then, the encoded items "1**", "11*" and "112" can be regarded as a set of distinguish items occurring on different level.

A set $X \subseteq I$ is called a *k*-itemset, if it contains *k* items at the same level. If $X \subseteq Y$, it is said that *X* occurs in t_j or t_j contain *X*. The support vale of *X*, denoted as s^X, is the ratio of *X*'s occurrence to the total number of transactions in database. Then, the support of X is used to define the concept of multi-level frequent itemsets as follows.

Definition 1: An itemset *X* is called *a multi-level frequent itemset* if *it contains encoded items at the same level of the concept hierarchy and its support is no less than the user-given support threshold.*

From above definition, the problem of mining multi-level frequent patterns is the task of discovering a complete set of frequent patterns on all the levels of concept hierarchy under user-given support thresholds. However, the setting of support thresholds is a difficult task. Then, we proposed to mine the *N*-most interesting frequent itemsets without any support threshold which can be defined as follows.

Definition 2: An itemset *X* is called *a N-most interesting multi-level frequent itemset* if *it contains at least two encoded items at the same level of the concept hierarchy and there is no more than N – 1 itemsets at the same level of X having support more than that of X.*

Thus, the mining of *N*-most interesting multi-level frequent patterns is to mine *N* itemsets at each level of concept hierarchy that have highest value of support under user-given the number of desired pattern *N*.

3 *NMLFP*: An Efficient Method for Mining *N*-most Interesting Multi-level Frequent Patterns

An efficient method for mining *N*-most interesting multi-level frequent patterns, called *NMLFP*, is described in this section. The proposed method is based on the concept of relative data item taxonomic together with mining *N*-most frequent itemsets. The proposed method applies *FP-tree* structure to capture the content of database and it consists of three phases: (*i*) construct a *FP-tree* for lowest-level items with two scanning of transactional database, (*ii*) mine *N* itemsets (with highest support) from the *FP-Tree* and then store all of them into *N*-most multi-level table and (*iii*) create a *FP-tree* for higher-level *FP-tree* from the current lower-level *FP-tree*, respectively. The details of *NMLFP* are described below:

STEP 1. Create header table for all of lowest-level items and then scan each transaction of *TDB* in order to collect occurrence frequency/support of each item into the header table. Lastly, sort the header table by support descending order.

STEP 2. Build *FP-tree* at lowest-level in the same manner as [2] in which each transaction is scanned, trimmed and sorted in the same order as the header table. Then, the sorted items in the scanned transaction are sequentially inserted into *FP-tree* with frequency to be 1 (if there exists a node of any regarded item in the *FP-tree*, its frequency is updated by one.)

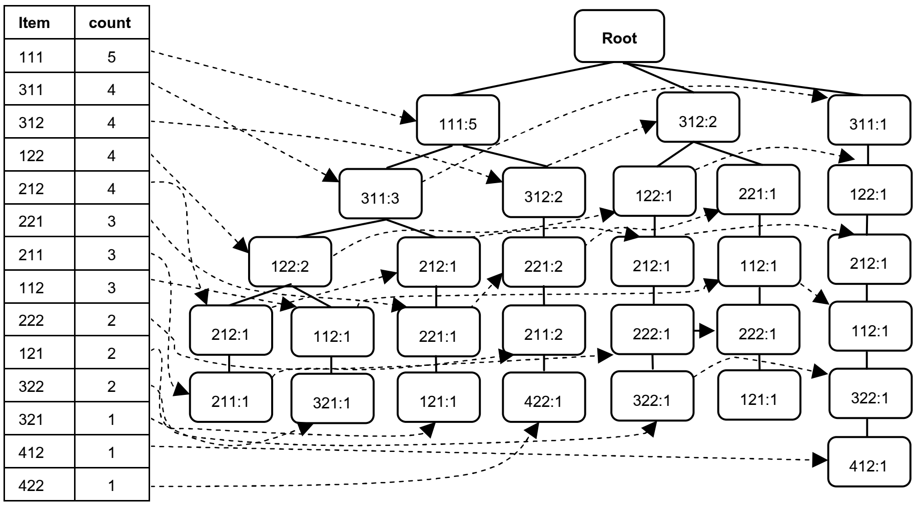
For illustration, we use an example with the encoded transactions shown in Table 1. Let the number of required results *N* for each level be 5 and the number of concept hierarchy level be 3. Our task is to discover the 5 itemsets with highest support from the three level of concept hierarchy level.

Table 1. The encoded transactional database

TID	Items
T01	111, 121, 212, 221, 311
T02	111, 211, 221, 312
T03	111, 122, 211, 212, 311
T04	112, 122, 212, 311, 322, 412
T05	112, 121, 221, 222, 312
T06	111, 112, 122, 311, 321
T07	111, 211, 221, 312, 422
T08	122, 212, 222, 312, 322

With the first and second scan of the encoded database on step 1 and 2, we get the header table and FP-tree for all of the lowest-level items as shown in Fig. 2.

STEP 3. Generate *N*-most frequent itemsets from the constructed *FP-tree* by (i) creating a table for storing a set of *N*-most interesting itemsets during mining process namely *N-most table*, (ii) applying the top-down traversal technique to consider items in the header table (consider from the second to the last item in the header table), (*iii*) traversing all the paths of the considered item *X* in the *FP-tree* in bottom-up manner in order to collect support of items appearing with *X* (consider only the items having higher position than that of *X* in the header table). If the support of an item *Y* occurring with *X* is greater than that of the N^{th} itemset in the *N-most table*, the N^{th} itemset is removed and the itemset '*XY*' is inserted into the *N-most table*. If there are itemsets '*XY*' (more than one), where *Y* is an item occurring with *X*, inserted into the *N-most table*, a *small FP-tree* for *X* with occurring with all items *Y* is created, (*iv*) traversing the *small FP-tree* of *X* by using *COFI-tree* traversal technique[8] to generate high-support itemsets with larger size.

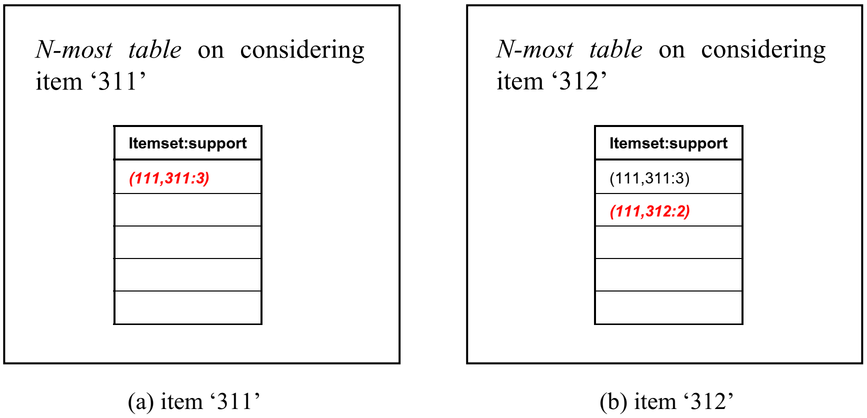


Item	count
111	5
311	4
312	4
122	4
212	4
221	3
211	3
112	3
222	2
121	2
322	2
321	1
412	1
422	1

Fig. 2. *FP-tree* for lowest-level items (level 3)

On step 3, *NMLFP* starts to consider the second item in the header table that is item '311', then all the paths of item '311' is traversed with the considering bottom items of the paths. We get {('111, 311':3), ('311':1)} and then we know that item '111' appear together with '311' three times. The support of itemset '111,311' is compared with the N^{th} itemset in the *N-most table*. Fortunately, there is none of itemset in the *N-most table*. Then, the itemset '111, 311' is inserted into the *N-most table* (as shown in Fig. 3(a)). Next, item '312' (the 3rd item in the header table) is considered. With the traversal of all paths we got {(111,312:2), (312:2)} and know that '111' appears together with the considered item twice. The support of '111, 312' is compared with the N^{th} itemset in the *N-most table*. Since there is only one itemset in the *N-most table*, the itemset ‘111, 312’ is inserted in to the *N-most table* by support descending order (as shown in Fig. 3(b)).

With the considering and traversing of item ‘122’, we gain {(111, 311, 122:2), (312, 122:1), (311, 122:1)}. Then, the itemsets ‘111, 122’, ‘311, 122’ and ‘312, 122’ with support 2, 3 and 1 are inserted into the *N-most table*. In this case, there is more than one itemsets of ‘122’ inserted into *N-most table*, and then a *small FP-tree* of ‘122’ with its header table is created as shown in Fig. 4.



N-most table on considering item ‘311’

Itemset:support
(111,311:3)

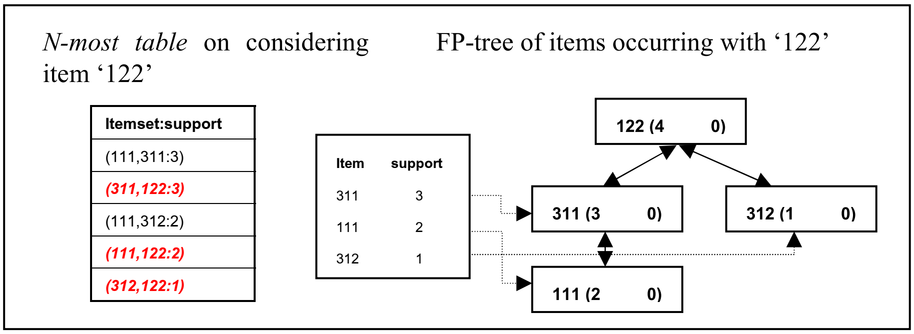
(a) item ‘311’

N-most table on considering item ‘312’

Itemset:support
(111,311:3)
(111,312:2)

(b) item ‘312’

Fig. 3. *N-most table* under the considering of items ‘311’ and ‘312’



N-most table on considering item ‘122’

Itemset:support
(111,311:3)
(311,122:3)
(111,312:2)
(111,122:2)
(312,122:1)

FP-tree of items occurring with ‘122’

Item	support
311	3
111	2
312	1

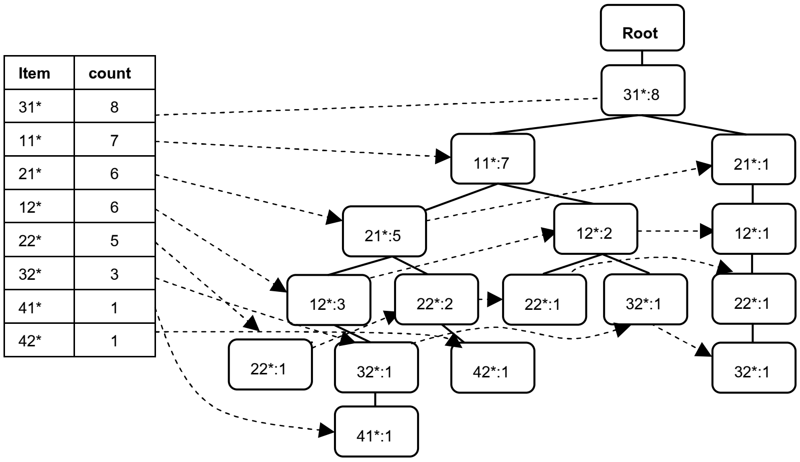
Fig. 4. *N-most table* and *a small tree* during the considering of item ‘122’

Next, the small tree of ‘122’ is traversed to get support of long itemsets. From Fig. 4, we can observe that itemset ‘111, 311, 122’ appear together twice. Then, its support is compared with the N^{th} itemset in the *N-most table* (*i.e.* itemset ‘312, 122’ with support 1). Since support of itemset ‘111, 311, 122’ is greater than that of ‘312,122’, the itemset ‘312, 122’ is eliminated and then the itemset ‘111, 311, 122’ is inserted into the *N-most table* by support descending order.

By repeating these processes on all items in the header table, we gain *N* itemsets with highest support at lowest level contained in the *N-most table* (as shown in the leftmost table in Fig. 6).

STEP 4. Construct a new *FP-tree* for higher-level items from the current *FP-tree* (low-level *FP-tree*) by (*i*) creating header table for all higher-level items, (*ii*) traversing each branch in the low-level *FP-tree* by encoding the last digit of item to be '*' (for example, item '111' is encoded to be '11*', or item '12*' is encoded to be '1**') in order to collect support of all higher-level item into the header table (if there is a high-level item appearing more than once in the considered branch, we will count only the highest support value), (*iii*) sorting the header table by support descending order, (*iv*) traversing each branch in the low-level *FP-tree* again to collect high-level items appearing in the considered branch (collect only one with highest support, if there is an item having multiple appearances in the considered path) and then sorting the items by the order of header table, (*v*) creating a path for the collected higher-level items into the higher-level *FP-tree*, and (*vi*) removing the regarded branch from the low-level *FP-tree*.

On step 4, the leftmost path of the lowest-level FP-tree (of Fig.2) is traversed and encoded to be {11*:5, 31*:3, 12*:2, 21*:1, 21*:1} (in this case the item '21*' appears twice, we only consider one with highest support). Then, the supports of all items in the path are used to update into the header table. Then, all of paths in the *FP-tree* are scanned, encoded and used to update support values of the header table. After considering all branches, the header table is ordered by support descending value (as shown in the left of Fig. 5). Next, all branches is scanned again. For the leftmost path, {111:5, 311:3, 122:2, 212:1, 211:1} is converted into {31*:3, 11*:3, 21*:1, 12*:1} and then sequentially inserted into the higher-level *FP-tree*. After traverse all branches of the lowest-level *FP-tree*, we gain the higher-level *FP-tree* (*FP-tree* of level 2) and its corresponding header table as shown in Fig. 5.

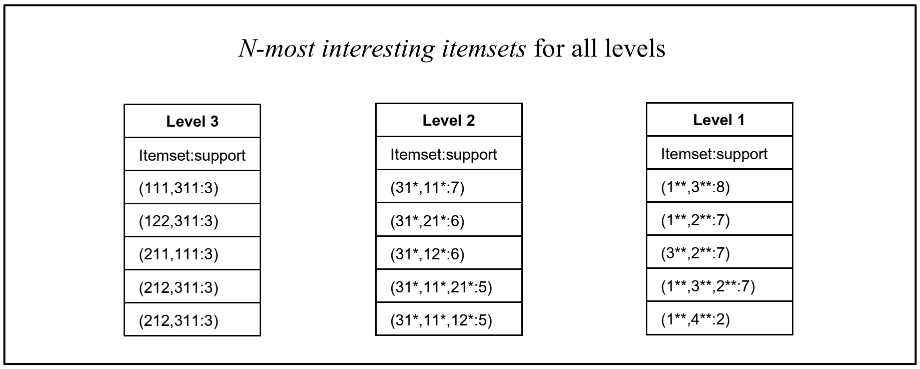


Item	count
31*	8
11*	7
21*	6
12*	6
22*	5
32*	3
41*	1
42*	1

Fig. 5. *FP-tree* for higher-level items (level 2)

STEP 5. Repeat step 3 and 4 on the new generated higher-level *FP-tree* until the highest level.

In our example, after creating and mining *N*-most interesting itemset from the *FP-tree* of all levels, the results of 5-most itemsets for all levels are shown as Fig. 6.



N-most interesting itemsets for all levels

Level 3	Level 2	Level 1
Itemset:support	Itemset:support	Itemset:support
(111,311:3)	(31*,11*:7)	(1**,3**:8)
(122,311:3)	(31*,21*:6)	(1**,2**:7)
(211,111:3)	(31*,12*:6)	(3**,2**:7)
(212,311:3)	(31*,11*,21*:5)	(1**,3**,2**:7)
(212,311:3)	(31*,11*,12*:5)	(1**,4**:2)

Fig. 6. *N-most table* for all levels of concept hierarchy

4 Performance Evaluation

We here study the performance of our proposed method by performing several experiments and then comparing with a modification of *mining multiple-level frequent itemsets algorithm, FPM-T* [6]. To conduct experiments, the value of *N* is set in range of [50, 2,000] and the support threshold at each level for *FPM-T* is assigned corresponding to the support of the N^{th} itemset at each level of *NMLFP* in order to get the same set of results. Both algorithms were implemented in JAVA. All the experiments are performed on 2.5 GHz Intel Core i5 and with 4GB main memory.

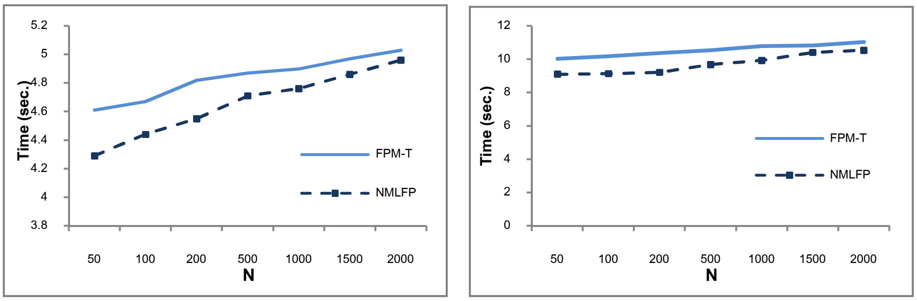
As shown in Table 2, two datasets, *T10I4D100K* and *T20I6D100K* with 4 concept hierarchical levels are used. *T10I4D100K* contains 100,000 transactions of size 10 in average, the potential maximal large itemsets of size 4, the number of node in the highest level is 8 and the number of fan-outs from level to level are 5, 5 and 5, respectively. Meanwhile, *T20I6100K* also contains 100,000 transactions of size 20 in average, the potential maximal large itemsets of size 6, the number of node in the first level is 15 and the number of fan-outs from level to level are 6, 3 and 4, respectively.

Table 2. Characteristic of datasets

Database	|*T*|	|*I*|	#node at level 1	Fan-outs 1 to 2	Fan-outs 2 to 3	Fan-outs 3 to 4
DB1	10	4	8	5	5	5
DB2	20	6	15	6	3	4

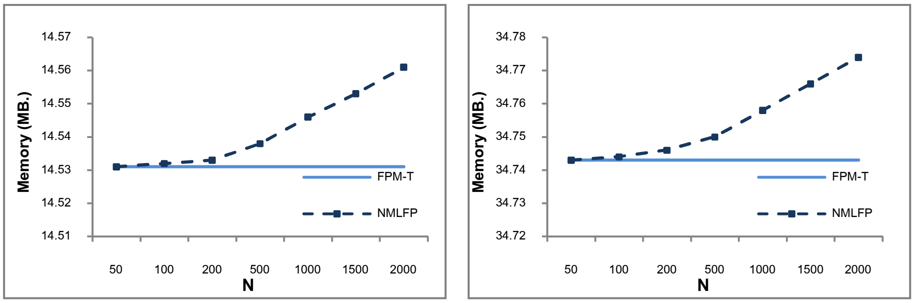
4.1 Execution Time

We consider the execution time of the both algorithms with all processes for mining the results. As shown in Fig. 7, the runtime of *NMLFP* is mainly efficient than that of *FPM-T* since *NMLFP* can take advantages from top-down traversal to mine results (*i.e.* firstly consider itemsets with highest support). This can help us to quickly gain *N* itemsets with highest support and raise the support of the N^{th}-itemset which is used as the criterion to cut-down the search space. Consequently, it slightly increases as the value of *N* increases. Whenever the value of *N* increases, we have to consider and discover more itemsets to be the results. Then, we have to take more computational time as well.



(a) T10I4D100K (b) T20I6D100K

Fig. 7. Execute time on (a) T10I4D100K and (b) T20I6D100K



(a) T10I4D100K (b) T20I6D100K

Fig. 8. Memory usage of NMLFP and FPM-T

4.2 Memory Consumption

To evaluate memory consumption of the both algorithms, we consider all components for mining the results including header table, and FP-tree. As indicated in Fig. 8, we can observe that *NMLFP* consumes a bit more memory than *FPM-T* due to *NMLFP* have to maintain a set of *N* most interesting itemsets for each level during mining process. Then, *NMLFP* uses a bit more memory than *FPM-T* for all cases. However, the gap of different is not significant. In all experiments, the gap of memory consumption is less than 1MB.

5 Conclusion

In this paper, we study the problem of multi-level frequent itemsets mining and then try to avoid difficulties from the setting of suitable support thresholds. Therefore, we propose the alternative approach to mine *N*-most interesting multi-level frequent itemsets without support threshold. This can discover the most *N* frequent itemsets for each level. To mine such itemsets, we introduce an efficient *NMLFP* algorithm based on the concept of FP-tree with single scan and top-down traversal to quickly generate the results. Our performance studies show that the proposed *NMLFP* algorithm achieves high performance in comparing with the previous approach. *NMLFP* is effective and efficient on both the small and large and it is also linearly scalable on the value of *N*.

References

1. Agrawal, R., Srikant, R.: Fast Algorithms for Mining Association Rules. IBM Almaden Research Center (1994)
2. Han, J., Pei, J., Yin, Y.: Mining Frequent patterns without candidate generation. SIGMOD Rec. 29(2) (2000)
3. Han, J., Fu, Y.: Discovery of Multi-Level Association Rules from Large Databases. In: 21st VLDB Conference on Very Large Data Base, Switzerland, pp. 420–431 (1995)
4. Lee, Y., Hong, T., Wang, T.: Multi-level fuzzy mining with multiple minimum supports. In: Expert Systems with Applications, pp. 459–468 (2008)
5. Hong, T., Huang, T., Chang, C.: Mining Multiple-level Association Rules Based on Pre-large Concepts. In: Data Mining and Knowledge Discover in Real Life Application, pp. 187–200. In Tech (2009)
6. Eavis, T., Zheng, X.: Multi-level frequent pattern mining. In: Zhou, X., Yokota, H., Deng, K., Liu, Q. (eds.) DASFAA 2009. LNCS, vol. 5463, pp. 369–383. Springer, Heidelberg (2009)
7. Mohammad, E., Osmar, R.: COFI-tree Mining: A New Approach to Pattern Growth with Reduced Candidacy Generation. In: Workshop on Frequent Itemset Mining Implementations (FIMI 2003) in Conjunction with IEEE-ICDM (2003)
8. Ngan, S., Lam, T., Wong, R., Fu, A.: Mining N-most interesting itemsets without support threshold by the COFI-tree. Int. J. Bus. Intell. Data Mining, 88–106 (2005)
9. Amphawan, K., Lenca, P., Surarerks, A.: Mining top-*k* Periodic-Frequent Pattern from Transactional Databases without Support Threshold. In: Papasratorn, B., Chutimaskul, W., Porkaew, K., Vanijja, V. (eds.) IAIT 2009. CCIS, vol. 55, pp. 18–29. Springer, Heidelberg (2009)
10. Amphawan, K., Lenca, P., Surarerks, A.: Mining top-*k* regular-frequent itemsets using database partitioning and support extimation. Expert Systems with Applications 39, 1924–1936 (2012)
11. Amphawan, K., Lenca, P.: Mining top-*k* frequent-regular patterns based on user-given trade-off between frequency and regularity. In: Papasratorn, B., Charoenkitkarn, N., Vanijja, V., Chongsuphajaisiddhi, V. (eds.) IAIT 2013. CCIS, vol. 409, pp. 1–12. Springer, Heidelberg (2013)