A NOVEL ALGORITHM:

APRIORI-ANT

THE COMBINATION OF APRIORI ALGORITHM AND ANTS COLONY OPTIMIZATION ALGORITHM

by

Choo Ai Ling

A dissertation submitted
In partial fulfillment of requirements for the degree of
Master of Advanced Information Technology
(Data Mining and Intelligent System)

Faculty of Computer Science and Information Technology
UNIVERSITI MALAYSIA SARWAK
2005

Table of Contents

Acknowledgements i

Abstract ii

Chapter 1: Introduction 1

Chapter 2: Literature Review 13

2.1 Apriori algorithm 13

2.1.1 Summary 25

2.2 Ants Colony Optimization algorithm 27

2.2.1 Pheromone trail 27

2.2.2 Transition rule. 28

2.2.3 Summary 36

Chapter 3: Data Collection Analysis 39

3.1 Data Collection 39

3.2 Data Format 42

Chapter 4: Methodology 46

4.1 Apriori algorithm 46

4.1.1 Apriori algorithm Computation section 47

4.2 Ants Colony Optimization algorithm 54

4.2.1 Pheromone calculation 54

4.3 Generating the rule 58

Chapter 5: Experimental Result 64

5.1 Experimental: Processing Time 67

5.1.1 Total Pheromone Value and Number of Records 67

5.2 Experimental: Rule Generation 72

5.2.1 What influences the number of rules generated? 72

5.2.2 How rules are generated 78

5.3 From Generated-Text file to Expert System 82

5.3.1 Based on 'In-General' 83

5.3.2 Based on 'Burn-Area' 85

5.4 Manually-Checking Analysis 88

5.4.1 In Apriori algorithm 88

5.4.2 In-Apriori-Ant-Internal-Calculation 96

5.4.2.2 Pheromone Evaporation. 105

Chapter 6: Discussion 110

Chapter 7: Conclusion 114

Appendix A 116

Appendix B 117

Appendix C 118

Appendix D 119

Appendix E 120

Appendix F 121

Appendix G 122

Appendix H 123

Appendix I 124

Appendix J 125

Appendix K 127

Appendix L 130

Appendix M 131

References 133

Acknowledgements

I would like to express my gratitude to the following persons and organization who had been of great help to me when I was compiling this thesis:

Associate Prof. Narayanan Kulathuramaiyer of Unimas, the Faculty Dean of Computer Science and Information Technology, who had taught me in the area of Data-Mining, and who had inspired me to choose this area to make my research.

Mr. Lee from Cognitive Science Faculty of Unimas who taught me in the area of Intelligent System.

Mr. Sylvester Arnab, my supervisor, who had been a great mentor and a spiritual supporter all along during the process of my thesis.

Kuching General Hospital from which I gathered data for my thesis. Special thanks to Mr. Kho and his staff in the Record Unit in helping me to trace the useful patients' case notes from a vast number of records.

Abstract

This thesis explores an innovative technique to extract data from a vast number of patients' records and then generate them into useful rules that can be used in an Expert System. Material used for the study of this thesis is a set of Skin Burn data.

Data-Mining is a process of gathering and analyzing data sets (often large) to find suspected relationship between them and summarize them in novel ways which are understandable and useful to a data owner.

Two well-known algorithms, Apriori algorithm and Ants Colony Optimization algorithm, are used in this thesis.

Apriori algorithm is the best-known rules discovery technique in Data-Mining, but its disadvantage is that it allows only the frequent itemsets to survive and form another level of itemsets and this process may leave out some interesting rules. Besides, the pruning and joining action in this algorithm prolongs the rule generation process.

Ants Colony Optimization algorithm is a probabilistic technique that can be reduced to find a good and shortest path.

In this thesis, the design of Apriori-Ant algorithm is to make use of the strength of Ants Colony Optimization algorithm to shorten the rule generation process in Apriori algorithm. At the same time, the joining process in Apriori algorithm is still retained until Level-2. The count from Level-1 (1-itemset) and Level-2 (2-itemset) is used in the modified pheromone formula in Ants Colony Optimization algorithm.

Rules are generated based on the highest pheromone value after 1000 cycles of Pheromone Update and Pheromone Evaporation process. No interesting rule is removed from the list if it is an infrequent item in Apriori algorithm. Thus all of the items are treated equal in this novel technique.

The experiment is carried out in two main areas: Processing Speed and Number of generated rules. From the finding, pheromone value and the items generated in 1-itemset (Level-1) and 2-itemset (Level-2) do greatly affect the Processing Time.

Chapter 1: Introduction

An Expert System is a system which dispenses expert advices and guidance on certain matters or problems. Thus, a Skin-Burn Expert System has to obtain expert knowledge from skin specialists and convert such knowledge into rules. The required information or medical data can be obtained from the case notes of skin-burn patients.

Medical data from patients' skin-burn case notes are not transactional data used in mining associate rules. Transactional data are data which can be obtained from a collection of items purchased at a grocery store or supermarket. The purchased items that go into each basket are analyzed to generate association rules.

An association rule is an implication of the form $X \Rightarrow Y$, where X and Y are each a set of items which occur together in a significant number of baskets. *Support S* is the *percentage of the total transactions* in which X and Y *occur together. Confidence C* is the number of transactions in which X and Y *occur divided by the number of transactions* in which only X *occurs and expressed as a percentage.* An association rule is one where S and C meet some minimum threshold requirements referred to as the *Minimum Support* (*Min Sup*) and *Minimum Confidence* (*Min Conf*) respectively.

This project introduces a novel technique – Apriori-Ant algorithm. It is an experiment to use non-transactional data to mine frequent patterns, which are data from a vast number of patients' case records, and generate them into association rules to be used in an Expert System. The data is different from the transactional data obtainable from the supermarket or grocery store.

Apriori-Ant algorithm is based on Apriori algorithm and Ants Colony Optimization (ACO) algorithm. Apriori-Ant algorithm uses only two I/O scans over the database for the whole process, thus eliminating the need for generating the candidate items.

So far, there is still no research yet on how to generate association rules from non-transactional data, and it is still not discovered as to whether or not non-transactional data can produce any useful rules.

Apriori algorithm [1, 3 & 15] is the best-known rules discovery technique in Data-mining. Data-mining, also known as Knowledge Discovery in Database (KDD), is a process of analyzing data sets (often large) to find suspected relationship between them and summarize them in novel ways that are understandable and useful to the data owner.

Apriori algorithm is an algorithm that finds interesting relationships between data. It mines the frequent itemsets to form rules; which employs an interactive approach known as level-wise search, searching from one itemset to another throughout the whole database.

This algorithm is good as far as finding quality rules are concerned, but if the database is large, then there will be many frequent items and many possible combinations. The reason is that, each time the candidate generation process will make as many scans over the database as the number of combinations of items in the antecedent, which is exponentially large. Due to combination explosion, it may lead to poor performance when frequent pattern sizes are large.

Apriori algorithm needs repetitive I/O disk scans over the database to generate frequent items and this involves huge computation during the candidates generation.

Another concern is that the number of frequent patterns is very much dependent on the support threshold. If the threshold is set to low, the occurrence of frequent patterns would be high. But if the threshold set to high, the occurrence of frequent patterns would be low, and this may reduce the quality of generated rules.

There are two measurements in Apriori algorithm which are called *Support measurement* and *Confidence measurement.* These two measurements are used to determine the threshold used in each joining and pruning process. The threshold is used to determine which items are to be pruned and which are to be joined in the next step. The frequent sets are those that support sets which are greater than minimum support threshold. Items that occur very infrequently or below the minimum support threshold in the data set are pruned although they may produce interesting and potentially valuable rules later.

As these two measurements may result in some interesting rules being left out, they are known as 'selfish' measurements because it allows only the itemsets greater than the minimum support to 'survive' to proceed to the next step.

An ACO algorithm [7 & 15] is a metaheuristic approach based on parameterized probabilistic model or pheromone model. A probabilistic technique is adapted to solve computational problems that can be reduced to find a good and shortest path.

The literature review [8, 9, 16, 18 & 20] shows that ACO algorithm is best for solving computational problems. The features of the main process of ACO algorithm, which are best for finding a good and shortest path in least amount of time, can be used to replace the tedious joining-and-pruning process in Apriori algorithm.

Joining-and-pruning process is the core process in Apriori algorithm, but this process requires repetitive I/O disk scans. So, it seems that the core process in ACO algorithm can be a remedy for the main process in Apriori algorithm.

This research identifies two main problems:

1. Repetitive I/O Disk Scans.

   In an Apriori algorithm, the candidates generation (joining and pruning action) takes up a lot of time and space while processing. Each cycle of candidates generation process requires full scan of database regardless of its size.

   To reduce repetitive I/O disk scans, either the candidates generation is to be reduced or the time-consuming candidates generation process be replaced.

   A lot of research has been done in this area. The proposed Apriori-Ant algorithm method is a technique which is basically intended for solving the problem of mining association rules, the large number of discovered patterns or rules [4], and the candidates generation process.

   Some researchers have used other similar techniques to replace candidates generation process such as the Trie method (similar to hash tree), HybridApriori; the combination of Apriori algorithm and AprioriTid algorithm, MLPFT. The Candidate and Frequent Pattern Tree technique such as COFI algorithm and generation process is replaced by a tree technique, but there are some limitations in each algorithm (more detail in Chapter 2 – Literature Review).

2. Huge computational cost required to generate frequent items.
   In Apriori algorithm, the candidate generation process involves not only the joining and pruning process, but also two other computational processes of Support measurement and Confidence measurement.

   The purpose of Support and Confidence calculation is for the next candidate generation. These values are to identify which itemsets have values that are above the threshold (Minimum Support value and Minimum Confidence value). Those items below the user-defined threshold are removed from the list which are called an infrequent itemsets.

   User-defined threshold may lead to poor quality rules being generated. There is no specific level to be taken as the best value to be used as the threshold in the process. If the threshold is set to low, then a high number of frequent patterns would be admitted, which thus requires much space for the searching, massive I/O, and high memory dependencies. If the threshold is set to high, it may not generate vast number of patterns, but then quality of generated rules may suffer.

   Each time a change is made on the user-defined threshold, the process needs to start over again. This makes Apriori algorithm technique time-consuming and requires high computational cost.

The pruning process may remove many rules that are useful. This brings us back to the problem of user-defined threshold in which the threshold that is set to high would generate a small amount of frequent patterns and thus may turn to be biased.

Some itemsets that are below the threshold may have good and interesting combination with other itemsets. For example, bread & margarine is combined with sugar & condensed milk. Sugar and condensed milk may not be interesting by themselves or with margarine, but it may be interesting if combined with both bread and margarine as some children like to have bread with some sugar spread on top of margarine or just bread with condensed milk.

The ideal way is to save all the itemsets and let an efficient process decide which itemsets are to be pruned and which are to be saved. This is thus an unbiased process.

The objective of this research is to find out how to improve an Apriori algorithm to solve the problems above and how an infrequent itemsets in Apriori algorithm can produce a series of useful association rules use in Medical Field. The proposed method is one which combines an Apriori algorithm with an ACO algorithm.

The research focuses mainly on the following questions:

1. How does an ACO algorithm replace the candidates generation process in Apriori-Ant algorithm to generate rules? Can this method solve the I/O overhead?

2. How efficient is the modification of measurement method (Support, Confidence and Lift) used in Apriori algorithm? Can it work well with an ACO algorithm? Can the

values generated by the Apriori algorithm measurement method be used in the state transition rule module in ACO algorithm which allows "*ants to move from one item (city) to another item (city)*"?

3. What affects the processing speed in Apriori-Ant algorithm? Is it the number of records or the number of rules generated? Or is there any other reason?

In this proposed algorithm, every itemset is treated equally, which means no itemset is pruned even if it is infrequent or below the user-defined threshold. The idea is not to prune the itemset during the Apriori algorithm but to calculate each of its support and confidence measurement. It is to make sure that the itemset which are below the minimum threshold will not pruned but to generate rules.

This design involves 5 parts in the process: Part one is to convert the data from text file into string data for processing. Part two is to generate the frequent itemsets and frequent counts. Part three is to perform the calculation. Part four is to join the rule based on the pheromone value. Part five is to write the rules into text file. All the processes can be found in Chapter 4 Methodology and Chapter 5 Experimental Result.

In part one, array [5 & 14] is introduced to handle the data read from the text file. All the string data will be stored into a string array for later process.

In part two, Apriori algorithm is introduced. This design adapts the Apriori algorithm method to find frequent itemsets and frequent counts for the respective itemset. But the process stops at the second level of finding, no pruning process is carried in this section.

The Apriori algorithm process is to find and store the frequent itemsets and frequent counts for 1-itemset and 2-itemset into the respective array.

Part three consists of two different computational sections. The first computation section is based on Apriori algorithm. In this section, one modified and two original measurement methods are used. The modified measurement method is Lift measure, and the original measurement methods are Support measure and Confidence measure that are used to assist in Lift measure.

The second computation section, which is based on ACO algorithm, substitutes the values that are generated from Apriori algorithm into the modified pheromone probability formulas. This value is used to decide which item is joined into the rule set and not to start a tour.

In part four, after probability pheromone values are generated, the rules generating process takes place. A tour is started from a randomly generated item in 1-itemset list to be matched with the items in 2-itemset list and only the item with the highest pheromone value would be joint into the rule set. This process is carried out until no more matching is found in 2-item sets. Modified Pheromone update and Pheromone evaporation is carried out in each cycle of process.

The last part is to write out the rules into a readable text file. A *connection* program is required to encode the text file so as to be readable by an Expert System.

The arrangement report is as below:

Chapter 2: Literature Review.

5 materials for Apriori algorithm and 5 materials for ACO algorithm are reviewed in this section.

For Apriori algorithm, the review is on techniques to improve its internal processing part. Research is done to improve efficiency of Apriori algorithm which includes techniques to replace the tedious pruning and joining process.

For ACO algorithm, the review is on techniques and how efficient ACO algorithm can be used. It also includes modified techniques used in ACO algorithm and techniques in ACO algorithm with Data Mining technique.

Chapter 3: Data Collection Analysis.

Data are manually collected from 190 patients' case notes of Kuching General Hospital and then converted into a digital form that can be used by this prototype system. This section shows how the process is carried out from data collected in manual format to digital format and how the digital format is then used in this prototype system.

Chapter 4: Methodology.

Methodology section covers the modification of Apriori algorithm, ACO algorithm and Rules Generation.

In Apriori algorithm, only the joining process is carried out which stops at Level-2 of the joining. No pruning process is performed. The output of Apriori algorithm is the 1-itemset generated at Level-1 with its count and also 2-itemset generated at Level-2 with its count.

In ACO algorithm, Lift-measurement is introduced into Apriori-Ant algorithm. The count at Level-1 and Level-2 is used in Lift measurement method to calculate the Pheromone value. And the Pheromone value is used in Pheromone Update and Pheromone Evaporation process in later part.

Methodology section also explains how rules are generated. Rules are generated based on the pheromone value calculated by the modified formula.

Chapter 5: Experimental Result.

Experiment is done in 4 areas: Processing Time, Rules Generation, Manual-Checking Analysis and From Generated-Text file to Expert System.

Processing Time: The experiment reveals that the influencing factors of processing speed are: *Total Pheromone value* and *Number of records.*

Total Pheromone value: The first finding of the experiment reveals that more pheromone requires more processing time.

Number of records: Apparently, more records should require more processing; but in the experiment, it is to find out how to reduce the process time regardless of number of records involved in the processing.

Rules Generation: The Experiment investigates two areas: What influences the number of rules generated and how are the rules generated?

It is found that the number of items in 1-itemset and 2-itemset influences the number of rules generated. The number of items in 1-itemset influences the number of items in 2-itemset. Items in 2-itemset indicate the number of rules generated by this prototype. From Generated-Text file to Expert System: The rules generated by this prototype system are stored into a text file. Before the rules can be used in an Expert System, another program is needed to re-arrange the rules into format that is compatible with Expert System.

Output from generated text file can be re-arranged in two areas: *General* and *Burn-Area.*

General Area: Rules are re-arranged in general basis. It is not based on any specific area like Burn-Area-*'Flame'*, or Body-Area-*'Finger'*.

Burn Area: Rules are re-arranged according to specific Burn-Area. For example, in *'Flame'* Burn-Area and *'Hot Oil'* Burn-Area, different treatments and medication are applied to each of them. Thus, in this section, rules are re-arranged based on the Burn-Area.

Manual Checking Analysis: This analysis is done manually in two sections: *In Apriori-algorithm* and i*n Apriori-Ant-Internal-Calculation.*

In Apriori-algorithm: The checking is carried out on the rules generated by the prototype with Apriori algorithm in manual form. And the results are compared with those generated by the prototype system.

In Apriori-Ant-Internal-Calculation: Checking is done on the internal calculation in Apriori-Ant algorithm using Microsoft Excel. The results are used to tally those produced by the prototype system.

Chapter 6: Discussion.

This research is to find out how Apriori-Ant algorithm solves the length pruning and joining process in Apriori algorithm; and how ACO algorithm is introduced into the middle part of Apriori-Ant algorithm.

Chapter 7: Conclusion.

Conclusion of this research.

Chapter 2: Literature Review

The aim of data-mining is to find the "***hidden gold***" in a data source; meaning to extract valuable information hidden in that data source. Such hidden valuable information may reveal a new business trend based on purchasing habits of consumers. Data-mining techniques are thus based on data retention and data distillation.

Apriori algorithm is one of the data-mining techniques which seeks to find and generate frequent itemsets by using support measurement to prune itemsets which are not frequent. Eliminating non-useful information in this way would save more space for more memory storage in a computer and thus improves its running time.

The objective of ACO algorithm is to solve Non-Polynomial problem (NP-problems) like route planning, scheduling, creating of timetables, etc. These problems need to be solved by using an approximation technique which is not an optimal solution to a given problem but is a solution that is good enough for that specific application.

2.1 Apriori algorithm

The basic concept of association rule mining is:

First, Apriori algorithm would analyze all the transactions in a dataset for each item support count. Let $J = \{i1, i2 \dots i\text{m})$ be a set of items. Let D, the task-relevant data, a set of database transactions where each transaction T is a set of items such as $T \subseteq J$. Any item that has a support count of less than the minimum support count threshold is removed from the candidate items list.

The frequently used measure methods in Apriori algorithm are Support and Confidence. To calculate a Support *(1)*, let A be a set of items. A transaction T is said to contain A if $A \subseteq T$. An association rule is an implication of the form $A \Rightarrow B$, where $A \subset J$, $B \subset J$ and $A \cap B = \phi$. The rule $A \Rightarrow B$ holds in the transaction set D with *Confidence c* if $c\%$ of the transactions in D that contain A also contains Y. The rule $A \Rightarrow B$ has supports s in the transaction set D if $s\%$ of transaction in D contains $A \cup B$. (i.e., both A and B).

$$Support(A \Rightarrow B) = P(A \cup B). \qquad (1)$$

For confidence method *(2)*, it is taken to be the probability, $P(A \cup B)$. The rule $A \Rightarrow B$ has confidence c in the transaction set D if c is the percentage of transactions in D containing A that also contain B. This is taken to be the conditional probability, $P(B \mid A)$.

$$Confidence(A \Rightarrow B) = P(B \mid A). \qquad (2)$$

Rule support and confidence are two measures of rule interestingness. Association rules are considered interesting if they satisfy both a minimum support threshold and a minimum confidence threshold. Such thresholds can be set by users or domain experts.

There are two-step processes in Apriori algorithm: Join step and Prune step.

In Join step, for example, in order to find L_k, a set of candidate k-itemsets is generated by joining L_{k-1} with itself. The join $L_{k-1} \bowtie L_{k-1}$ is performed, where members of L_{k-1} are joinable if the first (k-2) items are in common.

In Prune step, a scan of the database to determine the count of each candidate in C_k would result in the determination of L_k. C_k is a superset of L_k that is, its members may or may not be frequent, but all of the frequent k-itemsets are included in C_k.

Any *(k-1)*-itemset that is not frequent cannot be a subset of a frequent *k*-itemset. Hence, if any *(k-1)*-subset of a candidate k-itemset is not in L_{k-1}, then the candidate cannot be frequent either and so can be removed from C_k.

In the implementation of Apriori algorithm, the item that is less than the minimum support count will be removed from the candidate list. And items whose support count is greater than the minimum support count will be stored in one-candidate-itemsets list.

The remaining itemsets in one-candidate-itemsets list are joined to create two-candidate-itemsets list. The calculation of support count for two-itemsets is performed. Only if the support count is greater than the minimum support count, the remaining two-itemsets are joined to create three-candidate-itemsets. This process is iteratively performed until all item's support counts in the candidate-itemsets list are less than minimum support count.

All the candidate-itemsets generated with a support count greater than the minimum support count form a set of frequent itemsets. Apriori algorithm recursively generates all the subsets of each frequent itemset and creates association rules based on the subsets with a confidence greater than or equal to the minimum confidence. Thus a lot of interesting rules are pruned at this stage if it is an infrequent itemset.

$$confidence(A \Rightarrow B) = P(B \mid A) = \frac{support\_count(A \cup B)}{support\_count(A)} \qquad (3)$$

Where *support_count(A∪ B) (3)* is the number of transactions containing the itemsets *A∪B,* and *support_count(A) (3)* is the number of transactions containing the itemset *A*.

Thus it makes multiple passes over the database before generating a useful rule. It brings the problem to I/O overhead and high memory need.

This research shares some similarities with [1, 2, 6, 17 & 19]. Let us look into Apriori algorithm. Rakesh Agrawal and Ramakrishnan Srikant [1] are the Guru of Apriori algorithm. In [1] they proposed an AprioriHybrid algorithm to mining association rules in a faster way. AprioriHybrid algorithm is the combination of Apriori algorithm and Apriori-Tid algorithm.

Authors [1] focused on the problem of discovering association rules between items in a large database. Problem of mining association rules is to generate all association rules that have support and confidence greater than the user-specified *Minimum-Support* and *Confidence-Support.*

Let $I = \{i_1, i_2, ..., i_m\}$ be a set of literals, called items. Let D be a set of transactions, where each transaction T is a set of items such that $T \subseteq I$. In AprioriHybrid algorithm, associated with each transaction is a unique identifier, *TID*. Transaction T contains X, a set of some items in I, if $X \subseteq T$.

An association rule formed in AprioriHybrid algorithm is based on Apriori algorithm pattern. An implication of the form $X \Rightarrow Y$, where $X \subset I$, $Y \subset I$, and $X \cap Y = \phi$.

In AprioriHybrid algorithm, Apriori algorithm is introduced at the first level, it examines every transaction in the database. At this level, Apriori algorithm is used to generate the candidate itemsets, and the itemsets are to be counted in each pass and only the itemsets found large (frequent) in the database are survived to form the next level of set-of-itemsets.

After the candidate itemsets are generated, Apriori-Tid is introduced. In Apriori-Tid, the database is not used at all for counting the support of candidate itemsets after the first pass done in Apriori algorithm. If a transaction does not contain any candidate k-itemsets, then the set-of-itemsets will not have entry for this transaction thus the number of entries in set-of-itemsets maybe smaller than the number of transaction in the database.

Apriori algorithm is in the initial passes and later switch to AprioriTid when it expects the set-of-itemsets at the end of the pass will fit in memory.

The advantages of AprioriHybrid are that the size of the set-of-itemsets declines in the later passes; it scales linearly with the number of transactions, and the execution time decreases a little as the number of items in the database increases.

But the limitations are that, it incurs switching cost without realizing the benefits, and it suffers from the repetitive I/O disk scans. The candidates generation process still takes place in every cycle until the rules are generated. Although AprioriTid reduces the number of database, it still needs to scan in each candidate generation process.

In [2], the authors focus on algorithm running time and memory need. In this research [2], the problem for frequent itemsets mining is to find all frequent itemsets in a given transaction database, and this requires high running time and memory space as it needs to scan through the database more than one time.

As already mentioned, using Apriori algorithm requires repetitive I/O disk scans and memory space as it scans through database to find all the frequent itemsets. To reduce these requirements, the authors focus on the central data structure. They introduce a Trie Method to store not only candidates but also for frequent itemsets; which works like a hash-tree.

Tries are not only suitable to store and retrieve words but also applicable to any finite ordered sets. In the design, a link is labeled by an element of the set, and the Trie contains a set if there exists a path where the links are labeled by the elements of the set in increasing order.

The authors [2] modify the Support Count Methods with Trie. Support Count Method takes the transactions one-by-one. If a subset of an item is found to be a candidate, it would increase the support count of the respective candidate by one. This method does not generate all the respective subsets of certain transaction, but would perform early quits whenever possible. In this approach, Trie stores not only candidate but also frequent itemsets.

Support Count is done by reading transactions one-by-one to determine which candidates are contained in the actual transaction. Two simple recursive methods are introduced to solve the problem of finding candidates in a given transaction. And it is found that the running time difference is determined by the recursive step.

The Trie is built with frequency codes instead of original codes because we know exactly the frequency order after the first reading of the whole database.