A Scalable Algorithm for the Market Basket Analysis

Luís Cavique
ESCS, Instituto Politécnico de Lisboa, Portugal
e-mail: lcavique@escs.ipl.pt

Abstract. The market basket is defined as an itemset bought together by a customer on a single visit to a store. The market basket analysis is a powerful tool for the implementation of cross-selling strategies. Especially in retailing it is essential to discover large baskets, since it deals with thousands of items. Although some algorithms can find large itemsets, they can be inefficient in terms of computational time. The aim of this paper is to present an algorithm to discover large itemset patterns for the market basket analysis. In this approach, the condensed data is used and is obtained by transforming the market basket problem into a maximum-weighted clique problem. Firstly, the input dataset is transformed into a graph-based structure and then the maximum-weighted clique problem is solved using a meta-heuristic approach in order to find the most frequent itemsets. The computational results show large itemset patterns with good scalability properties.

Keywords: market basket analysis, frequent itemset mining, maximum-weighted clique

1. Introduction

This paper addresses the problem of finding market baskets in large databases [Berry and Linoff 1997]. Current database capacities associated with bar code technology and growth of the Internet has led to a huge collection of customer transaction data.

In Chris Anderson's latest book, "The Long Tail" [Anderson 2006], he explains why the future of retail business involves selling smaller quantities of more products. Anderson sums up his idea as the "98% rule" contrasting with the well known "80/20 rule". The "98% rule" means that in a statistical distribution of the products, only 2% of the items are very frequent and 98% of the items have very low frequencies, creating a long tail distribution. The long tail shape emerges in the new markets due to three factors: democratization of the tools of production, democratization of the tools of distribution and finally the connection between supply and demand based on online networks. The long tail distribution depends on the type of retailer: the physical retailer is limited by the store's size, corresponding to a short tail; then the online retailers, like Amazon.com, expanded the number of products, creating a longer tail; finally, the pure digital retailers, like Rhapsody that sells music on-line, working with no physical goods, have expanded the long tail even further. The emergence of the "98% rule" in the retail sector has made the software that works with many low-frequency items more relevant and appealing.

The performance of Apriori-like algorithms has greatly improved in computational time in the last decade, achieving very good scalability. However, the output has consistently been small market baskets. In the retail business, which deals with thousands of items, it is essential to discover larger baskets.

In this paper, we proposed a different approach to obtain large itemsets. The Similis algorithm performs in low time complexity using condensed data. This swift algorithm transforms the current problem into a graph, and searches for complete sub-graphs (or cliques) that are equivalent to market baskets.

In section 2 we define the market basket analysis and the state of the art algorithms for this problem.

In section 3 we present the Similis algorithm, which first of all transforms the input dataset into a graph-based structure, and then the new problem, the weighted clique problem, is solved using a meta-heuristic approach. Each maximum-weighted clique corresponds to a frequent itemset.

In section 4 the Similis algorithm is validated and the computational results are presented for one real dataset and other datasets from the Frequent Itemset Mining Implementations Repository.

Finally, in section 5 the conclusions point out the differences between the Apriori-like algorithms and the Similis algorithm and its advantages in retailing.

In this approach the Frequent Itemset Mining and the Graph Theory terminology are used. We will refer to the market basket (or k-itemset) whenever it is related to physical data. On the other hand, if it refers to condensed data the terms complete sub-graph (or clique) are used. In this work the keywords market basket, k-itemset, complete sub-graph or clique of size k, are equivalent.

2. The Market Basket

2.1 The definition of the problem

The input for the market basket analysis is a dataset of purchases. A market basket is composed of items bought together in a single trip to a store. The most significant attributes are the transaction identification and item identification. While ignoring the quantity bought and the price. Each transaction represents a purchase, which occurred in a specific time and place. This purchase can be linked to an identified customer (usually carrying a card) or to a non-identified customer.

The dataset with multiple transactions can be shown in a relational table (transaction, item). Corresponding to each attribute there is a set called domain. The table (transaction, item) is a set of all transactions $T=\{T_1, T_2, T_3, \ldots, T_n\}$ where each transaction contains a subset of items $T_k = \{I_a, I_b, I_c \ldots\}$.

To exemplify this problem, an instance with 5 items and 7 transactions is given in table 1. The domain(item) is equal to {a, b, c, d, e} and the domain(transaction) is equal to {1, 2, 3, 4, 5, 6, 7}.

Table 1- Sample data to illustrate the market basket analysis

date	customer	transaction	item
05-Jan	1001	1	b, c, d
05-Jan	1003	2	a, b
05-Jan	1004	3	a, c, e
07-Jan	1001	4	b, c, d, e
07-Jan	1005	5	a, c, d
07-Jan	1003	6	a, d
07-Jan	1006	7	b, c

Based on the attributes (transaction, item), the market basket will be defined as the N items that are bought together more frequently. Once the market basket with N items is known, we can move on to cross-selling. The next step is to identify all the customers having bought N−m items of the basket and suggest the purchase of some *m* missing items. In order to make decisions in marketing, the market basket analysis is a powerful tool supporting the implementation of cross-selling strategies. For instance, if a specific customer's buying profile fits into a identified market basket, the next item will be proposed.

The cross-selling strategies lead to the recommender systems. A traditional Recommender System is designed to suggest new products to frequent customers based on previous purchase patterns [Wang et al. 2004]. One of the first approaches, in order to find the market basket consists in using the Collaborative Filtering software developed by Net Perception that finds a "soul mate" for each customer [Hughes 2000]. A customer's "soul mate" is a customer who has identical tastes and therefore the same market basket. The software is installed in hundreds of companies. However, its disadvantage is that it merely compares two individuals and this does not allow an overall view. For example: customer X bought four books about Data Mining and a book about Cooking, and customer Y bought the same four books about Data Mining. Therefore, the software will suggest the Cooking book as the next item for customer Y, leading to possible mistakes.

2.2 The Apriori Algorithm

In opposition to the "soul mate" algorithm, the Apriori Algorithm [Agrawal and Srikan 1994, Agrawal et al. 1996] takes all of the transactions in the database into account in order to define the market basket. The market basket can be represented with association rules, with a left and a right side Left⇒Right. For instance, given an itemset {A,B,C} the rule {B,C}⇒{A} should be read as follows: if a customer bought {B,C} he would probably buy {A}. This approach was initially used in pattern recognition and it became popular with the discovery of the following rule: "on Thursdays, grocery store customers often purchase diapers and beer together" [Berry and Linoff 1997].

To evaluate the association rules two measures can be used, the support measure and the confidence measure. Let {A,B} be an itemset and the let A⇒B be the association rule. The support measure is equal to the relative frequency or the probability P(A∩B). The confidence measure is given by the conditional probability of B given A, P(B|A), which is equal to P(A∩B)/P(A). The rule "diapers=>beer" with a support measure of 50% and a confidence measure of 85% means that diapers and beer are bought together in 50% of the transactions and in 85% of cases those who buy diapers will also buy beer, i.e. P(beer|diapers)=85%.

The Apriori algorithm was implemented in commercial packages, such as Enterprise Miner from the SAS Institute [SAS 2000]. As input, this algorithm uses a table with purchase transactions. Each transaction contains its identification and the purchased items, as follows (transaction, item). Input parameters are defined as the maximum number of acquired items (max_k) and the minimum support (minsup) of a certain basket. The Enterprise Miner package from SAS Institute implemented this algorithm using max_k=4 and minsup=5% as default values. The minsup is of extreme importance in the algorithm since it will prune the useless branches in the search.

The minsup is a parameter used to control the combinatorial expansion of the exponential algorithm. As Apriori uses only a single value for the minsup, some basket sizes may be oversized while others may be undersized.

The first step of the Apriori algorithm generates sets of market baskets. I_k is defined as the set of frequent items with k items bought together. Firstly, the algorithm filters the items with a frequency that is higher than the minsup, generating I_1. In the following stages, for each I_k it generates the I_{k+1} candidates, such as $I_k \subseteq I_{k+1}$. For each I_{k+1} candidate, the algorithm removes the baskets, which are lower than the minsup. The cycle ends when it reaches I_{max_k}.

In the second step, the Apriori algorithm generates sets of market baskets and then generates association rules Left⇒Right. For each rule, the support measure and the confidence measure are calculated.

The outputs of the Apriori algorithm are easy to understand and many new patterns can be identified. However, the sheer number of association rules may make the interpretation of the results difficult. A second weakness of the algorithm is the computational times when it searches for large itemsets, due to the exponential complexity of the algorithm.

2.3 Apriori-like Algorithms Related Work

The Apriori algorithm has an exponential time complexity and several passes over the input table are needed. To overcome these handicaps some proposals have been made.

The Apriori algorithm performs as many passes over the data as the size of the itemsets. The Dynamic Itemset Counting, the DIC Algorithm, reduces the number of passes made over the data, using itemsets forming a large lattice structure [Brin et al. 1997]. DIC starts counting the 1-itemset and then adds counters to 2,3,4,…,k itemsets.

Thus, after a few passes over the data it finishes counting all the itemsets. Running DIC and Apriori, DIC outperformed Apriori in the majority of cases.

In the developing of recommender systems, i.e., systems that personalize recommendations of items based on the customer's preferences, in [Lin, Alvarez and Ruiz 2002] the authors present an algorithm that does not require the predefined minsup measure, allowing the marketeer to tune that measure. Having previously defined minsup a negative result can be expected by pruning either too many or too few itemsets.

FP-growth [Han, Pei, Yin 2000] is similar to several algorithms for frequent itemset mining that preprocesses the transaction databases, discarding the infrequent items. Currently it is one of the fastest approaches, since it condenses the transaction information into a tree structure. Computational results show that FP-growth scales better than the previous algorithms. Borgelt [2005] presents a C implementation of the FP-growth that outperforms well-known implementations in several instances. However, although the FP-growth achieves better scalability it still returns small itemsets.

3. The Similis Algorithm

To overcome the difficulty of finding large itemsets, we have developed a new algorithm — the Similis (meaning similar in Latin) due to the fact that a k-itemset is similar to a clique. This term was chosen just like Apriori, since they are both Latin names.

In this section, firstly we describe some graphs concepts required for the understanding of the algorithm. Secondly, the Similis algorithm is presented in two steps − the transformation step and the search step. Finally, the similarities and differences between this method and other methods are discussed.

3.1 Graph Concepts

A possible way to condense the data is by transforming it into a graph structure. A graph is a pair $G=(V,E)$ of sets satisfying $E \subseteq [V]^2$ where elements of E are 2-element subsets of V. The elements of V are the vertices (or vertexes, or nodes, or points) of the graph G(V,E) and the elements of E are its edges (or arcs, or lines).

Given the undirected graph G(V,E), then G1(V1,E1) is called a subgraph of G if $V1 \subseteq V$ and $E1 \subseteq E$, where each edge of E1 is incident in the vertices of V1. A subgraph G1 is said to be complete if there is an arc for each pair of vertices. A complete subgraph is also called a clique. A clique is maximal if it is not contained in any other clique. In the maximum clique problem the objective is to find the largest complete subgraph in a graph. The clique number is equal to the cardinality of the largest clique of the graph. If a weight is given to each edge, the subgraph is known as a weighted subgraph, and the weight of the subgraph is given by the sum of the weights of the edges, as follows:

$$\text{Clique\_weight} = \sum \text{weight}(i,j), \ \forall \ \text{edge}(i,j) \in \text{Clique} \qquad (1)$$

A clique can represent a common interest group. Given a graph representing the communication among a group of individuals in an organization, each vertex represents an individual, while edge (i,j) shows that individual *i* regularly communicates with individual *j*. Finding a common interest group where each individual communicates with all of the other group members, corresponds to finding a clique. In French "la clique" is defined as a closely knit group of individuals who share common interests. Finding the maximum clique means finding the largest common-interest group possible.

If a graph with weights in the edges is used, the highest weighted clique corresponds to the common-interest group whose elements communicate the most among themselves. This structure allows the representation of sets of elements that are strongly connected.

The Maximum Clique Problem is an important problem in combinatorial optimization with many applications, which include: market analysis, project selection and signal transmission [Berge 1991]. The Maximum Clique Problem is a hard combinatorial problem, classified as NP-Hard [Garey and Johnson 1979]. The interest in this problem led to the algorithm thread challenge on experimental analysis and algorithm performance promoted by Second DIMACS Implementation Challenge [Johnson and Trick 1996].

In Cavique, Rego and Themido [2002] the Primal-Tabu, the Dual-Tabu and the Primal-Dual-Tabu algorithms are presented to solve de Maximum Clique Problem. The Primal-Tabu is an extension of the algorithm developed by Soriano and Gendreau [1996]. To find the Maximum-weighted Clique with size k, the Primal-Tabu algorithm was adapted by adding the restriction related to the clique size and the objective function was changed.

Three possible moves were identified: the related neighborhood structures are N^+, N^-, and N^0 for the addition, removal and swap of a vertex of the subgraph. To avoid cycling in the iterative search process, a short-term memory is incorporated. The short-term memory is usually implemented using a Tabu list, made up of a set of (reverse) moves, which are forbidden for a certain number of iterations.

The original version of Primal-Tabu combines the neighborhood structures N^+, N^0 and N^-. At each step, one new solution S' is chosen from the neighborhood N(S) of the current solution S. If it is possible to increase the clique value, one vertex is added to the clique using N^+, or else try to swap vertices using the neighborhood structure N^0, otherwise the heuristic moves backwards removing one vertex using N^-. At each iteration the current solution S and the best solution found S* are updated whenever the clique value is increased.

The weighted version of the algorithm restricts the growth of $N^+(S)$ by using the condition $|S|<k$; the objective function was changed, instead of using the vertex number, the function for the clique weight is now used. The local search will run through the solution space searching for the maximum-weighted clique with a given dimension k. The Primal-Tabu Heuristic can be sketched as follows:

S is the current solution and S' the trial solution
S* is the best solution found
Tabu is the tabu list
$N^+(S)$, $N^-(S)$, $N^0(S)$ are the neighborhood structures

Primal-Tabu Meta-Heuristic for the Maximum-Weighted Clique

input: weighted graph G(V,E), size k;
output: maximum-weighted clique S* with size k;
initialize S, S* e Tabu;
    while not end condition
        if $N^+(S)\backslash Tabu \neq \emptyset$ and $|S| < k$ choose the best S´;
        else if $N^0(S)\backslash Tabu \neq \emptyset$ choose the best S´;
            else choose the best S´ in $N^-(S)$ and update Tabu;
        update S←S';
        if Clique_weight(S) > Clique_weight(S*) then S*←S;
    end while;
return S*;

The heuristic finds the maximum-weighted clique with time complexity $\Theta(N^3)$ where N is number of items.

Graph based structures are being increasingly used in Data Mining. For instance, in order to solve a clustering problem, M. Zaki [2000] decomposes the original search space (or lattice) into smaller pieces (or sub-lattices). Two ways for achieving the decomposition are proposed: the prefix-based and the maximal-clique-based partition. A maximal-clique itemset is a set of items that is not included in other itemset cliques. This only occurs in sparse graphs. For dense graphs, there is an overlap among the cliques. To solve this problem, the author points out the "weak maximal cliques". This itemset clique clustering technique, using equivalent classes, outperformed the computational results of the other approaches.

3.2 The Algorithm Description

To find the market basket patterns, i.e. the most frequent itemsets, the Similis algorithm is described in two steps – the data transformation step and the search step. For the first step, the input is the table T(transaction, item) and the output is a weighted graph G(V,E). For the second step, the input is the graph G(V,E) and the size of the market basket *k* and the output is the market basket with k-items. For the same transformation step, the search step can run more than once, depending on the market basket dimension one is looking for.

Firstly, if the number of items is too large, a filter can be used to discard the infrequent 1-itemsets, equivalent to the minsup cut. The weighted graph G(V,E) is generated, using the information of the 2-itemset frequency. In the graph G(V,E) the vertex set V represents the number of items. The weighted edge $(i,j) \in E$ represents the times that item *i* and item *j* were bought together in the transactions. The procedure to load the graph is the following: given a transaction, create a clique with the given itemset and add one to each edge of the clique in the graph. The process is repeated

for all the transactions, returning the adjacent matrix of the weighted graph G, presented in table 2.

In the second step, to find the maximum-weighted clique that corresponds to the most frequent market basket, we adapted the Primal-Tabu Meta-heuristic presented in section 3.1. The main algorithm can be sketched as follows:

The Similis Algorithm

STEP 1 – Data Transformation
input: support measure, table T(transaction, item);
output: weighted graph G(V,E);

- Discard the infrequent 1-itemset using a filter;
- Generate graph G(V,E) using the 2-itemset frequency;

STEP 2 – Finding the Maximum-Weighted Cliques
input: weighted graph G(V,E) and size k;
output: weighted clique S of size *k* that corresponds to the most frequent k-itemset;

- Find in G(V,E) the clique S with *k* vertexes with the maximum weight, using the Primal-Tabu Meta-heuristic.

In the first step, the condensed data is created, in such a way that the second step can run as many times as needed, altering the market basket size, thus showing one of the advantages of having condensed data.

In the transformation step, to create the graph, the time complexity is $\Theta(N)$ where N is the number of transactions. In the search step, to find the maximum-weighted clique with size k, we use a heuristic approach with time complexity $\Theta(N^3)$ where N is number of items.

To illustrate the Similis algorithm, two numeric examples are presented as follows:

Numeric Example 1 - Problem Transformation:
Using the data present in table 1, the transactions set is T={T1,T2,T3,T4,T5,T6,T7} where transaction T1={b,c,d}, T2={a,b}, T3={a,c,e}, T4={b,c,d,e}, T5={a,c,d}, T6={a,d} and T7={b,c}. The problem transformation returns de adjacent matrix of the weighted graph G(V,E) in table 2.

Table 2 - Adjacent matrix of the weighted graph G(V,E)

G(V,E)	b	c	d	e
a	1	2	2	1
b		3	2	1
c			3	2
d				1

Numeric Example 2 - Searching for the Maximum-Weighted Clique:
Following the previous numeric example, based on the weighted graph, a chart is presented in figure 1, where only the edges which weigh more than 1 are shown, in order to clarify the figure.

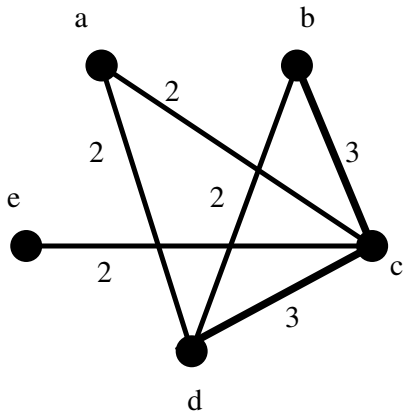

Figure 1 – Weighted Graph where the 3-itemset (b, c, d) corresponds to the most weighted clique

Two cliques with three vertexes stand out from the graph: the clique (b,c,d) and (a,c,d). For the clique (b,c,d) the sum of the edges is equal to 8 and for clique (a,c,d) the value 7 is obtained. The most weighted clique is (b,c,d) that corresponds to the most frequent 3-itemset that occurs twice.

3.3 Similarities and Differences

The Similis algorithm is based on the conjecture that a clique is similar to a k-itemset. However, some differences can be found. The 3-itemset (a,c,d) in figure 1 occurs twice, while in the clique (a,c,d) the weight of edge (c,d) is equal to 3, so the problem transformation creates a false weighted itemset in (c,d). The drawback of this approach is the fact that in the final graph some specific weighted cliques can appear and they don't correspond to any k-itemset transaction. Nevertheless, the graph structure offers a different type of information.

The soul-mate algorithm compares a k-itemset with an identical k-itemset of a single customer. On the other hand, Apriori-like algorithms find all the exact k-itemsets. In a more general way, the Similis algorithm, mixes all the preferences in the graph G(V,E) going further than a single comparison or an exact itemset count, by creating a pattern.

Data mining can be defined as the science and the art of pattern discovery. A pattern is a shape, a form, a template or more abstractly a model. A pattern is created by the combination of repeated single templates, resulting in a final form with a specific character — in nature we can easily identify a crystal, a tree or a fish pattern and fractals are artificial patterns produced by mathematical models. To discover a pattern

is to discern a relevant template, with the desired structure in a general set. In our approach, it seems that the information given by the weighted graph is richer than the frequency of a market basket with k-items, since it includes the combined information of different market baskets, as shown by the pattern definition. Each weighted clique is a pattern that was created by the combination of repeated single itemsets and the Primal-Tabu Meta-heuristic is able to discern the desired templates.

When dealing with problems that are hard to solve, approximate algorithms are commonly used nowadays. We can find many examples of this in the fuzzy sets, genetic algorithms and neural networks. Using this kind of approach, we think that the transformation of data into a graph can be useful.

4. Computational Results

In the validation of an algorithm some choices must be made such as the datasets, the computational environment and the performance measures. The chosen performance measures are the accuracy of the Similis algorithm and the computational times associated to the minsup. The FP-growth algorithm implemented by Borgelt [2005] will be used in the computational result comparisons.

To validate the Similis algorithm a real dataset named frozen-food and datasets from the Frequent Itemset Mining (FIM) Implementations Repository were used. The real dataset contains information related to the distribution of frozen food items throughout Portugal by the Nestle enterprise.

Table 3 – Datasets

dataset	size (KB)	nr item	average # item/trans	max # item/trans
frozen-food	169	159	8	59
retail	4,156	18,000	11	77
accidents	34,679	468	34	54

The three datasets in table 3 were chosen keeping in mind different sizes and different number of items. The same table shows the size in KB, the number of items, the average number of items per transaction and the maximum number of items per transaction for each dataset. Given this data, it is important to study large market baskets within the 10 - 70 item range.

The computer program was written in C language and the Microsoft Visual C++ compiler was used. The computational results were obtained from a Pentium 4 CPU 2.8 GHz with 240 MB of main memory running in Windows XP.

4.1 Performance Measure 1 : Similis Accuracy

The FP-growth algorithm has one possible measure: the support measure. The Similis algorithm retrieves only the clique weights as the performance measure, so the use of the support measure seems pertinent to obtain the accuracy of Similis algorithm.

For each algorithm the top five solutions were reached. For each market basket with k items, found by the Similis algorithm, an accuracy measure is required. For the FP-growth and Similis algorithms the solutions are ordered by decreasing support measure.

The Accuracy function is given by the average support of the top five solutions found by the Similis algorithm, divided by the average support of the top five solutions found by the FP-growth algorithm, that is:

$$\text{Accuracy} = \frac{\sum \text{Similis support}}{\sum \text{FPgrowth support}}.100 \qquad (2)$$

In table 4, the best five 10-itemsets from both algorithms are presented for the retail dataset. Although the items found in both solutions are very similar, they must be assessed by the support measure. In this case we obtain an accuracy of 75% by applying equation (2).

Table 4 – The best five 10-itemsets from FP-growth and Similis

Similis			**FP-growth**	
clique	**sup**		**itemset**	**sup**
17 61 79 83 84 95 96 97 98 99	1.2%		58 79 82 83 84 95 96 97 98 99	1.5%
17 58 61 83 84 95 96 97 98 99	1.1%		58 79 83 84 95 96 97 98 99 100	1.4%
17 61 74 83 84 95 96 97 98 99	1.0%		58 79 83 84 87 95 96 97 98 99	1.4%
17 61 82 83 84 95 96 97 98 99	1.0%		79 83 84 87 95 96 97 98 99 100	1.4%
17 61 83 84 87 95 96 97 98 99	1.0%		79 82 83 84 95 96 97 98 99 100	1.4%
	5.3%			**7.1%**

4.2 Performance Measure 2: Computational Time and Minsup

In table 5 the computational results are presented for the datasets frozen-food, retail and accidents. Firstly, we run the Similis algorithm, with different itemset sizes: 5, 10, 15, 20 and 25. Then using the FP-growth algorithm, we search for the adequate minsup measure that returns the k-itemset.

In the Apriori-like algorithms the desired k-itemset and the computational time are dependent on the minsup measure. So, some trials must be carried out. In these trials we use the minsup measure in following the decreasing order: 10%, 5%, 1%, 0.5%, 0.1%, 0.05%, 0.01%, 0.005% and 0.001%.

Table 5 – Computational Results

	Similis					FP-growth	
dataset	**filter**	**vertex**	**itemset**	**time (sec.)**	**accuracy**	**minsup**	**time (sec.)**
frozen-food	0.0%	159	5	33	95%	1%	1
	0.0%	159	10	34	75%	1%	1
	0.0%	159	15	35	-	0.1%	>900
	0.0%	159	20	39	-	-	-
retail	0.5%	222	5	309	100%	0.1%	1
	0.5%	222	10	331	0%	0.01%	3
	0.5%	222	15	352	0%	0.005%	6
	0.5%	222	20	381	-	0.001%	>900
accidents	0.0%	468	10	102	91%	10%	37
	0.0%	468	15	154	89%	10%	27
	0.0%	468	20	200	99%	5%	119
	0.0%	468	25	244	-	1%	>900

The computational times in the Similis algorithm are steady when the k-itemset increases, showing a good scalability. On the other hand, the FP-growth doesn't show the same scalability, since it is difficult to find market baskets containing more than 20 items.

For the datasets frozen-food and accidents the accuracy measure performs well. However, for the retail dataset the results are not adequate. The Similis algorithm finds patterns of large itemsets that don't correspond to real large itemsets. FP-growth ca use very low support measures that Similis can't reach for the retail dataset with 18,000 items. On the other hand, Similis can obtain large k-itemsets in every datasets.

In order to understand these computational results, table 6 shows the results of the best five 5-itemsets and 10-itemsets given by Similis and FP-growth for the retail dataset. The results of the 5-itemset are identical for both algorithms, whereas the results for the 10 –itemsets are completely different.

We can also assert that the results obtained by the FP-growth for the 10-itemsets don't include any item from the 5-itemsets. With such a solution it would be impossible to implement a cross-selling strategy, since the notion of market basket is not presented. On the other hand, the Similis results are steadier allowing the implementation of a cross-selling strategy.

As already referred the Similis algorithm returns itemset patterns that result from the combination of many market baskets, going further than the exact itemset counting.

Table 6 –The best five 5-itemsets and 10-itemsets given by Similis and FP-growth

Similis		FP-growth	
5-clique	**sup**	**5-itemset**	**sup**
38 39 41 48 170	0.5%	32 38 39 41 48	0.5%
32 38 39 41 48	0.5%	38 39 41 48 170	0.5%
38 39 41 48 110	0.4%	36 38 39 41 48	0.4%
36 38 39 41 48	0.4%	38 39 41 48 110	0.4%
38 39 41 48 89	0.2%	32 38 39 48 170	0.2%
10-clique	**sup**	**10-itemset**	**sup**
32 36 38 39 41 48 65 89 170 475	0.0%	587 1012 1933 2363 3431 3836 4269 4524 7869 10804	0.0% 11
32 36 38 39 41 48 65 89 170 310	0.0%	150 587 1012 1933 2363 3431 3836 4524 7869 10804	0.0% 11
32 36 38 39 41 48 65 89 170 237	0.0%	237 587 1012 1933 2363 3431 3836 4524 7869 10804	0.0% 11
32 36 38 39 41 48 65 89 170 225	0.0%	150 587 1012 1933 2363 3431 3836 4269 7869 10804	0.0% 10
32 36 38 39 41 48 65 89 110 170	0.0%	237 587 1012 1933 2363 3431 3836 4269 7869 10804	0.0% 10

5. Conclusions

The main contribution of this paper is the discovery of large itemset patterns with good scalability in large datasets. The emergence of the "98% rule" in the retail sector has made the software that works with large patterns more relevant and appealing. In this approach, the condensed data is obtained by transforming the market basket problem into a maximum-weighted clique problem.

There are three important variables that influence the computational time: the number of transactions, the basket size and the number of items. The time complexity of the original Apriori algorithm is dependent on the number of items, the number of transactions and the market basket size. Using a condensed tree data structure, the FP-growth is dependent on the number of items and the market basket size, and it scales better than Apriori. Finally, the time complexity of the Similis algorithm is only dependent on the number of items (that correspond to the number of vertexes of the graph) showing a good scalability. The number of transactions is not relevant for the FP-growth and Similis algorithm due to the transformation into a tree or into a graph problem, allowing the processing of a huge number of transactions.

In the Similis algorithm the computational results show the discovery of large itemset patterns with good scalability. In order to find any k-itemset, we use a heuristic approach with time complexity $\Theta(N^3)$ where N is the number of items. To validate the Similis algorithm real datasets and datasets from the FIM literature were used. In future work, we would like to improve the performance of the Primal-Tabu Meta-heuristic to deal with a larger number of vertexes. FP-growth can use very low support measures that Similis can't reach in datasets with thousands of items. On the other hand, Similis can obtain very large k-itemsets.

In table 3, the average number of items per transactions and the max number of items per transaction points to the importance of studying large market baskets patterns. The Similis algorithm finds large purchase patterns that sometimes doesn't correspond to

the exact itemset count, but express the combination of several purchases, with a good scalability properties.

References

R. Agrawal, R. and R. Srikan, "Fast algorithms for mining association rules", Proceedings of the 20th international conference on very large data bases, 478-499, 1994.

R. Agrawal, H. Mannila, R. Srikant, H. Toivonen and A. Verkamo, Fast Discovery of Association Rules, in Advances in Knowledge and Data Mining, U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth and R. Uthurusamy Eds, MIT Press, 1996.

C. Anderson, The Long Tail: why the future of business is selling less of more, Hyperion Books, 2006.

C. Berge, Graphs, 3rd edition, North-Holland, 1991.

M. Berry and G. Linoff, Data Mining Techniques for Marketing, Sales and Customer Support, John Wiley and Sons, 1997.

C. Borgelt, An Implementation of the FP-growth Algorithm, Workshop Open Source Data Mining Software, OSDM'05, Chicago, IL, 1-5.ACM Press, USA, 2005.

S. Brin, R. Motwani, J.D. Ullman and S.Tsur, Dynamic Itemset Counting and Implication Rules for Market Basket Data, in Proceedings of the 1997 ACM SIGMOD Conference, Tucson, Arizona, 1997, pp.255-264.

L. Cavique, C. Rego and I. Themido, A Scatter Search Algorithm for the Maximum Clique Problem, in Essays and Surveys in Meta-heuristics, C. Ribeiro and P. Hansen Eds, Kluwer Academic Publishers, 2002, pp. 227-244.

M.R. Garey and D.S. Johnson, Computers and Intractability: a Guide to the Theory of NP-Completeness, W.H. Freeman and Company, New York, 1979.

J. Han, J. Pei, Y. Yin, "Mining frequent patterns without candidate generation", Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, Dallas, Texas, United States, 1-12, 2000.

A.M. Hughes, Strategic Database Marketing, McGraw-Hill, 2000.

D.S. Johnson and M.A. Trick Eds, Clique, Coloring and Satisfiability, Second Implementation Challenge DIMACS, AMS, 1996.

W. Lin, S.A. Alvarez and C. Ruiz, Efficient Adaptive-Support Association Rule Mining for Recommender Systems, Data Mining and Knowledge Discovery, 6, Kluwer Academic Publishers, 2002, pp.83-105.

SAS Software, Enterprise Miner Documentation, SAS Institute, 2000.

P. Soriano and M. Gendreau, Tabu Search Algorithms for the Maximum Clique, in Clique, Coloring and Satisfiability, Second Implementation Challenge DIMACS, D.S. Johnson and M.A. Trick Eds, AMS, 1996, pp. 221-242.

Y. Wang, , Y-L. Chuang, M-H. Hsu, Huan-Chao Keh, "A personalized recommender system for the cosmetic business", Expert Systems with Applications, 26, 427-434, 2004.

M. Zaki, Scalable Algorithms for Association Mining, IEEE Transactions on Knowledge Engineering, vol. 12, 3, May/June 2000, pp. 372-390.