P: ISSN NO.: 2394-0344 RNI No.UPBIL/2016/67980 VOL-3* ISSUE-8* (Part-1) November- 2018
E: ISSN NO.: 2455-0817 *Remarking An Analisation*

Application of Association Mining Algorithms to Market Basket Analysis for Decision Making

Shishu Pal Singh
Assistant Professor,
Dept. of Computer Science,
Government Post Graduate College,
Noida, Gautam Buddh Nagar,
Uttar Pradesh, India

Abstract

When a customer buys some items together from a store during his single visit is termed as market basket and the items are termed as itemset. Analysis of such market basket is very helpful for the implementation of cross – selling strategies. There are many algorithms to find market basket. Some of them are better in some way to others. This paper gives a comparative study of various algorithms of finding association rules from market basket data.

In this paper we discuss the problem of analyzing market-basket data and discussed several important contributions. In the beginning, we have given an algorithm to find large itemsets using fewer passes over the data than classic algorithms, and also using fewer candidate itemsets than other methods based on sampling. Here given the idea of item reordering, this can improve the low-level efficiency of the algorithm.

Then, we give a new way of generating "implication rules". Thereafter we discussed a new approach which uses data-condensed structures. In this approach, the condensed data is generated by converting the market basket problem in a maximum weighted clique problem. Initially, the provided data set 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.

Keywords: Item Sets, Market basket, Association Rule, Implication Rule, Graph.

Introduction

In this paper we discuss the problem of identifying market baskets in huge databases. Current database capacities associated with bar code technology and increased use of the Internet has led to a huge collection of customer transaction data.

Now companies from different sectors such as insurance, banking, airlines and telecommunications have become much more customers oriented than never. To get the customer's personal data there are two key data sources using the customer's personal data and the product – oriented data. In order to collect customer's social, demographic, personality, geographic or lifestyle data, costly surveys are required. While, product-oriented data, about the frequency and the quantity of a particular item which a customer buys is already exists in the companies' database. In order to setup customer relationship strategy one needs to find out that who the best customers are, how they respond to a campaign and are capable to predict the next purchasing item each customer will buy next time, thus implementing a cross-selling strategy.

In second section we define the market basket problem and discuss the Apriori algorithm that gives solution to this problem. As this algorithm has a non-polynomial time complexity, we discuss related work that tries to overcome this drawback.

In third section we have discussed an efficient algorithm – DIC (Dynamic Itemset Counting), which break the database into parts and introduce the concept of partial parallel computing to increase the efficiency of the algorithm.

In next section we gives a swifter algorithm - Similis, which first of all convert the provided data set 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 quasi-most-frequent itemset.

At the end, in last section the related works in the area are given with little description.

During this paper we refer to the market basket (or itemset) whenever it is related to physical data, on the other hand, if it refers to condensed data the terms graph-based structure (or clique) are used.

Market Basket Analysis

Definition

The file having records of set of purchases is given as input to the market basket analysis. A market basket is constituted of items purchased together in a single visit to a store. The most important fields are the customer identification and item identification, not concerned with the quantity bought and the price. Each transaction represents one purchase, which occurred at a specific time and place, and may be associated to an identified customer (usually having a card) or to a non-identified customer.

Definition 1

The record file with multiple transactions can be represented in a relational database table T(customer, item). Corresponding to each attribute there is a non-empty set called as domain. The domain (customer) = {1, 2, 3,. . . n} and the domain(item) = {a, b, c, . . . , z}. The table T(customer, item) can be read as the set of all customer transactions Trans ={t1, t2, t3, . . . , tk } where each transaction has a subset of items tk = {ia, ib, ic . . . }. The relational table T(customer, item) may also be read as the set of item-clientele Itc={i1, i2, i3, . . . } where each item-clientele contains a subset of customers ik={c1, c2,c3,. . . }.

On the base of attributes (customer, item), the market basket can be defined as the N items which are purchased together more frequently. Once the market basket with N items is known, we can shift to cross-selling. Further we identify all the customers having bought N-m items of the basket and then suggest them the purchase of some m missing items. In making decisions in marketing applications, the market basket analysis is a important tool supporting the implementation of cross-selling strategies. For example, if any specific customer's buying habits fits into a known market basket, the next item will be proposed.

Large itemsets finding Algorithms

Much research has focused on deriving efficient algorithms for finding large itmesets (step 1). The most well-known algorithm is Apriori which, as all algorithms for finding large itemsets, relies on the property that an itemset can only be large if and only if all of its subsets are large. It proceeds level-wise. First it counts all the 1-itemsets and finds counts which exceed the threshold – the large 1-itemsets. Then it combines those to form *candidate* (potentially large) 2-itmesets, counts them and determines which are the large 2-itemsets. It continues by combining the large 2-itemsets to form candidate 3-itemsets, counting them and determining which are the large 3-itemsets and so forth.

Implication Rules

Our contribution to functionality in market basket analysis is *implication rules* based on conviction, which we believe is a more useful and intuitive measure than confidence and interest. Unlike confidence, conviction is normalized based on both the antecedent and the consequent of the rule like the statistical notion of correlation. Furthermore, unlike interest, it is directional and measures of these two features, implication rules can produce useful and intuitive results on a wide variety of a data. For example, the rule *past active duty in military* $\Rightarrow$ *no service in Vietnam* has a very high confidence of 0.9. Yet it is clearly misleading since having past military service only increases the chances of having served in Vietnam. In tests on census data, the advantages of conviction over rules based on confidence or interest are evident.

Apriori Algorithm

In contradiction to the earlier algorithm, the Apriori Algorithm [Agrawal et al. 1996] takes all of the transactions in the database into consideration in order to define the market basket. The market basket can be represented with association rules, with a left and a right side L $\Rightarrow$ R. For example, given an itemset {A,B,C} the rule {B,C} $\Rightarrow$ {A} should be read as follows: if a customer bought {B,C} then there are high chances that he would buy {A} too. 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 find the association rules two measures can be used - the support measure and the confidence measure. Let {A,B} is an itemset and the let A $\Rightarrow$ B be the association rule. The support measure is equal to the relative frequency or 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 initial step of the Apriori algorithm gives sets of market baskets. I_k is defined as the set of frequent items with k items bought together. First, the algorithm finds the items with a frequency that is higher than the min sup, generating I1. In the following steps, for each I_k it generates the I_{k+1} candidates, such as I_k I_{k+1}. For each I_{k+1} candidate, the algorithm removes the baskets, which are lower than the min sup. The cycle ends when it reaches I_{max_k}.

In the second step, the Apriori algorithm gives sets of market baskets and then generates association rules L $\Rightarrow$ R. For each rule, the support measure and the confidence measure get calculated. In order to implement the cross-selling strategy the data analysts choose, firstly, the dimension of the basket, secondly, they choose the rules with the highest support measure. Finally, those having highest confidence measure are chosen, among those with the highest support measure.

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. Another weakness is the computational times, due to the exponential complexity of the algorithm.

Let L_K be the set of large k-itemsets. For example, L_3 might contain {{A,B,C}, {A,B,D}, {A,D,F}, …}. Let C_k be the set of candidate k-itemsets; it is always the superset of L_k.

Here is the algorithm:

Result := Ø;
K := 1;
C_1 = set of all 1-itemsets;
While $C_k \neq$ Ø do
create a counter for each itemset in C_k;
forall transactions in database **do**
increment the counters of itemsets in C_k which occur in the transaction;
L_k := All candidates in C_k
Which exceed the support threshold;
Result := *Result* U L_k ;
C_{k+1} := all k + 1-itemsets
Which have all of their k-item subsets in L_k
K := k + 1;
End

Thus, the algorithm performs as many passes over the data as the maximum number of elements in a candidate itemset, checking at pass k the support for each of the candidates in Ck. The two important factors which govern performance are the number of passes made over all the data and the efficiency of those passes.

DIC Algorithm

To overcome both issues of Apriori algorithm we discuss Dynamic Itemset Counting (DIC), the algorithm which reduces the number of passes made over the data while keeping the number of itemsets which are counted in any pass relatively lesser as compared to methods based on sampling.

DIC discussed the high-level issues of when to count which itemsets and is a substantial speedup over Apriori, particularly when Apriori requires many passes.

The algorithm which counts complete large itemsets must find and count all of the large itemsets and the minimal small itemsets (that is, all of the boxes and circles). The DIC algorithm, given here, marks itemsets in four different possible ways:

1. Solid Box – confirmed large itemset – an itemset we have finished counting that exceed the support threshold.
2. Solid Circle – confirmed small itemset – an itemset we have finished counting that is below the support threshold.
3. Dashed Box – suspected large itemset – an itemset we are still counting that exceeds the support threshold.
4. Dashed Circle – suspected small itemset – an itemset we are still counting that is below the support threshold.

The DIC algorithm work as follows:

1. Solid box represents the empty itemset. All the 1-itemset are represented with dashed circles. All other itemsets are unmarked. (See Figure 3.)
2. Read M transactions. We experimented with values of M ranging from 100 to 10,000. For each transaction, increment the respective counters for the itemsets marked with dashes.
3. if a dashed circle has a count that exceeds the support threshold, turn it into a dashed square. If any immediate superset of it has all of its subsets as solid or dashed square, add a new counter for it and make it a dashed circle. (See Figure 4 and 5.)
4. If a dashed itemset has been counted through all the transactions, make it solid and stop counting it.
5. If we are at the end of the transaction file, rewind to the beginning. (See Figure 6.)
6. If any dashed itemsets remain, go to step 2.

This way DIC starts counting just the 1-itemsets and the quickly adds counters 2,3,4,… ,k-itemsets. After just a few passes over the data (usually less than two for small values of M) it finishes counting all the itemsets. Ideally, we would want M to be as small as possible so we can start counting itemsets very early in step 3. However, step 3 and 4 incur considerable overhead so we do not reduce M below 100.

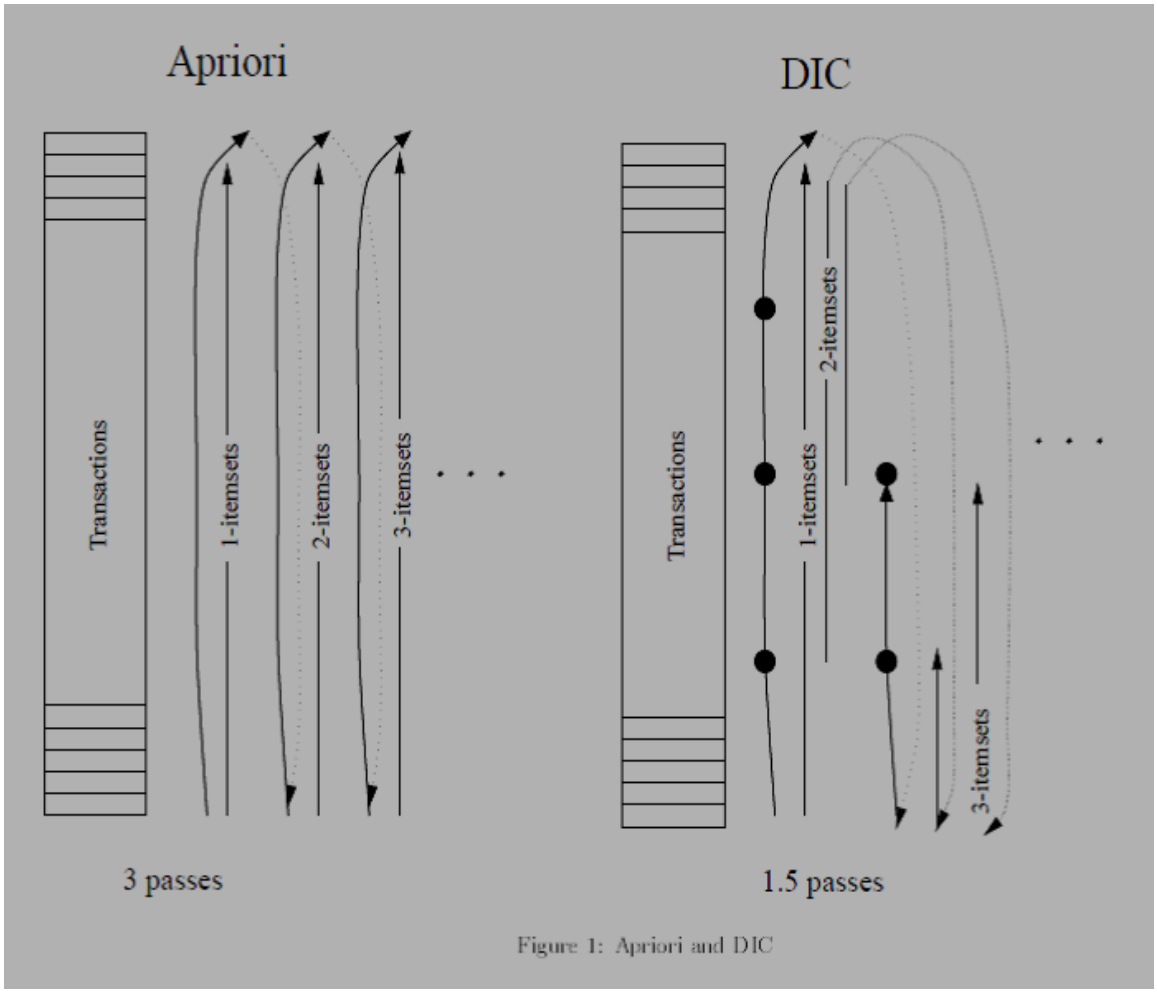

Figure 1: Apriori and DIC

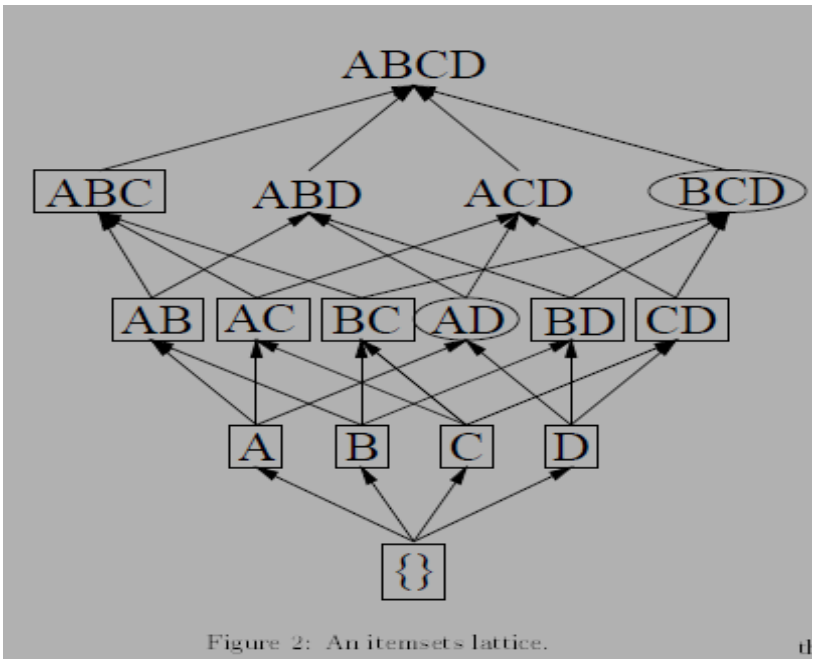

Figure 2: An itemsets lattice.

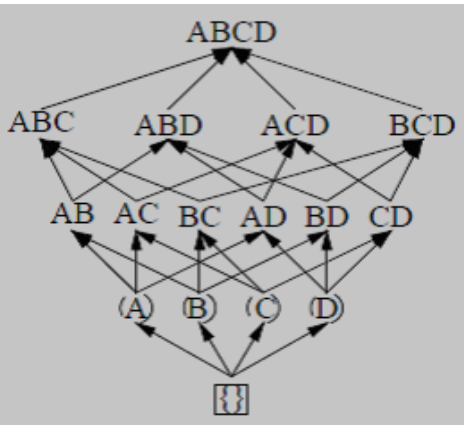

Figure 3: Start of DIC algorithm.

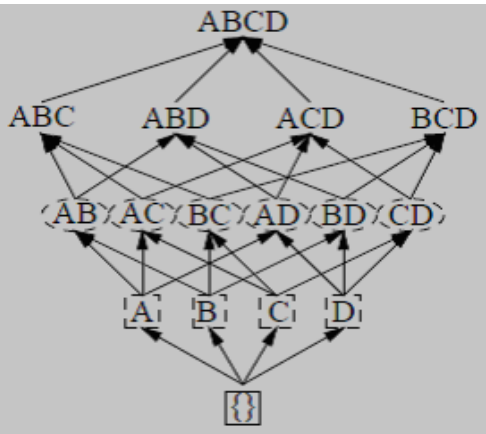

Figure 4: After M transactions.

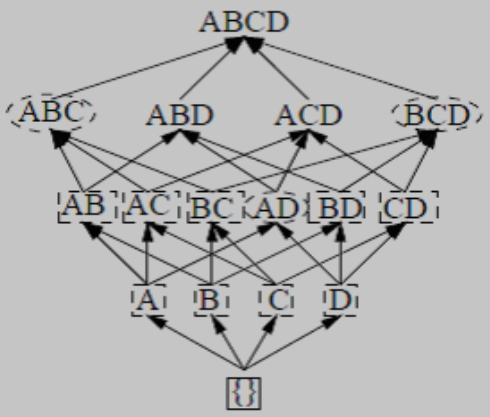

Figure 5: After $2M$ transactions.

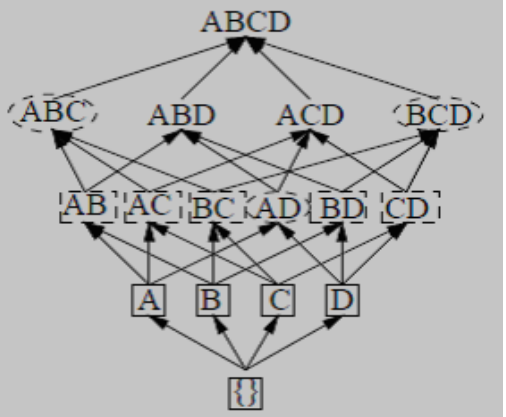

Figure 6: After one pass.

The Data Structure

The implementation of the DIC algorithm requires a data structure which can keep track of many itemsets. In particular, it must support the following operations:

1. Add new itemsets.
2. Maintain a counter for every itemset. When transactions are read, increment the counters of those active itemsets which occur in the trasaction. This must be very fast as it is the bottleneck of the whole process.
3. Maintain itemset states by managing trasitions from active to counted (dashed to solid) and from small to large (circle to square). Detect when these transitions should occur.
4. When itemsets do become large, determine what new itemsets should be added as dashed circles since they could now potentially be large.

The data structure used for DIC is exactly same as the hash tree used for Apriori with some extra information stored at each node. It is a tree having following properties. Each itemset is sorted by its items. Every itemset we are counting or have counted has a node associated with it, as do all of its prefixes. The empty itemset is the root node. All the 1-itemset are attached to the root node, and their branches are labled by the item they represent. All other itemsets are attached to their prefix containing all but their last item. They are labeled by that last item.

Significance of DIC

The main benefit of DIC is its performance. If the data is fairly homogeneous throughout the file and the interval M is reasonably small, this algorithm usually makes on the order of two passes. This makes the algorithm reasonably faster than Apriori which must make as many passes as the maximum size of a candidate itemset. If the given data is not fairly homogeneous, the algorithm may run through it in a random order.

Similis Algorithm

The Similis algorithm finds the most frequent itemsets in two steps – data transformation and searching.

In the first step we input table T(customer, item) and get output a weighted graph G(V,E). In second step, input given is the graph G(V,E) and market basket size k and get output a market basket with k items. As per the market basket dimensions

required, the last step of searching can run more than once.

After first step a weighted graph G(V,E) is developed depending on the similarities of the items. In the graph G(V,E) the vertex set V denotes the itemset in the market basket and weighted edge (i,j)2E indicate the similarity between item i and item j. the two items are same if they were bought together in number of transactions.

At the end, to get the clique with maximum weight which leads to most frequent market basket, Primal – Tabu Meta – heuristic is used. The Similis Algorithm is as follows [1]:

The Similis Algorithm

STEP 1 - Data Transformation
input: table T(customer, item)
Generate graph G(V,E) using the similarities between items
output: weighted graph G(V,E)
STEP 2 – Find clique with maximum weight
input: weighted graph G(V,E) and size k

Find in G(V,E) the clique S with k vertexes with the maximum weight, using the Primal- Tabu Meta-heuristic.
output: weighted clique S of size k that correspond to the most frequent market basket with k items.

Related Work

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 2, 3, 4, . . . , k itemsets. Thus, after a few passes over the data it _nishes counting all the itemsets. Running DIC and Apriori, DIC outperformed Apriori in the majority of cases.

In [Aggarwal, Wolf and Yu 1999] a method for indexing market basket data for similarity search is discussed. The index structure requires the partition of the set of all k-itemsets into subsets. They create a graph so that each node corresponds to an item, and for each pair of items a weight is added, which is the inverse of the support measure. Finally, the set of items is divided into k-sets. This algorithm shows good scalability with an increase in the number of transactions.

Just like the DIC algorithm, the MARC algorithm [Liu, Lu and Lu 2001] avoids several passes over the databases. MARC algorithm will use the summarized cluster information. The algorithm analyzes the similarities between transactions and creates clusters of similar transactions.

In the GCTD algorithm [Chen et al. 2002], the concepts of similarity relationships and the clustering problem appear together in order to discover connected components in an undirected graph.

The condensed data representation is extremely useful [Jeudy and Boulicaut 2002], taking into account that the Apriori algorithm has a better performance using sparse data rather than using highly correlated data. The latter is considered difficult or even intractable.

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 min sup measure. Having previously defined min sup, a negative result can be expected, by cutting down either too many or too few itemsets.

In order to obtain frequent market baskets in reduced computational times, in the next section we present the Similis Algorithm [Cavique e Themido 2001] [Cavique 2002]. This algorithm reuses some of the mentioned strategies, such as the reduction of passes over the database, the reduction of the number of parameters (e.g. min sup) and the aggregate measures (e.g. similarity measures).

Some important relevant work was done by Toivonen using sampling. His technique was to sample the data using a reduced threshold for safety, and then count the necessary itemsets over the whole data in just one pass. However, this pays the added penalty of having to count more itemsets due to the reduced threshold. This can be quite costly, particularly for databases like the census data. Instead of being conservative, our algorithm bravely marches on, on the assumption that it will alter come back to anything missed with little penalty.

Conclusion

Here in the beginning of the paper, importance of Market Basket Analysis is explained thereafter Apriori algorithm is presented. As in Apriori, many passes of data is occurred which leads to exponential time complexity of this algorithm and becomes its main drawback. Such algorithm is suitable when using the less items or sparse data but as we use correlated data the performance degrades remarkably. If we decrease the items number in the provided data using the min sup parameter, due to min sup independency from data table it may gives unpredictable data reductions. In the last Apriori gives large number of associative rules, out of which very less rules are utilized during cross – selling strategies.

We found that the DIC algorithm, particularly when combined with randomization provided a significant performance boost for finding large tiemsets. Item reordering did not work as well as we had hoped. However in some isolated earlier tests it seemed to make a big difference. We suspect that a different method for determining the item ordering might make this technique useful. Selecting the interval M made a big deference in performance and warrants more investigation. In particular, we may consider a varying interval depending on how many itemsets were added at the last checkpoint.

There are a number of possible extensions to DIC. Because of its dynamic nature, it is very flexible and can be adapted to parallel and incremental mining.

After discussing all the drawbacks of Apriori algorithm, we suggest the Similis algorithm due to its better computational complexity.

References

1. *L. Cavique, Graph Based structure for the Market Basket Analysis, Investigacao Operacional, 24 [2004], 233 - 246.*
2. *L. Cavique and I Themido, A New Algorithm for the Market Basket Analysis, Internal Report CESUR-IST, UTL, Portugal, 2001.*
3. *C.C. Aggarwal, J.L. Wolf and P.S. Yu, A New Method for Similarity Indexing of Market Basket Data, in Proceedings of the 1999 ACM SIGMOD Conference, Philadelphia PA, 1999, pp.407-418.*
4. *E. Balas, W. Niehaus, Optimized Crossover-Based Genetic Algorithms will be the Maximum Cardinality and Maximum Weight Clique Problems, Journal of Heuristics, Kluwer Academic Publishers, 4, 1998, pp. 107-122.*
5. *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.*
6. *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.*
7. *H. Toivonen. Sampling large databases for association rules. Proc. of the Int'l Conf. on Very Large Data Bases (VLDB), 1996.*
8. *R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In Proceedings of the 20th VLDB Conference, Santiago, Chile, 1994.*
9. *R. Agrawal, T. Imilienski, and A. Swami. Mining Association Rules between Sets of Items in Large Databases, Proc. Of the ACM SIGMOD int'l Conf. on Management of Data, pages 207-216, May 1993.*
10. *R. Agrawal, T. Imilienski, and A. Swami. Data base Mining: A Performance Perspective, IEEE Transactions on Knowledge and Data Engineering, 5(6):914-925, December 1993.*