Expert Systems with Applications 41 (2014) 1928–1936

Contents lists available at ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

Extending market basket analysis with graph mining techniques: A real case

Ivan F. Videla-Cavieres *, Sebastián A. Ríos

University of Chile, Department of Industrial Engineering, Business Intelligence Research Center (CEINE), Santiago, Chile

ARTICLE INFO

Keywords:
Market basket analysis
Graph mining
Retail
Product network
Big data
Overlap communities

ABSTRACT

A common problem for many companies, like retail stores, it is to find sets of products that are sold together. The only source of information available is the history of sales transactional data. Common techniques of market basket analysis fail when processing huge amounts of scattered data, finding meaningless relationships. We developed a novel approach for market basket analysis based on graph mining techniques, able to process millions of scattered transactions. We demonstrate the effectiveness of our approach in a wholesale supermarket chain and a retail supermarket chain, processing around 238,000,000 and 128,000,000 transactions respectively compared to classical approach.

1. Introduction

Over decades retail chains and department stores have been selling their products without using the transactional data generated by their sales as a source of knowledge. Recently – in the last two decades – companies started to use this data to discover information. In the 90's limited computational capabilities made the extraction of knowledge from millions of daily transactions unfeasible, and only analysis with simple models and reduced datasets were possible. In 1993, Agrawal (Agrawal, Imielinski, & Swami, 1993; Agrawal & Srikant, 1994) showed that many organizations were getting bigger databases with transactional data, consumer data, sales records, etc. Therefore, they proposed the Apriori algorithm (Agrawal & Srikant, 1994) for a large data set for those years.

Today, computational systems have evolved – both hardware and software – and have been implemented in all areas of companies (CRMs, ERPs, MRPs, Data Marts, Data Warehouses, ad hoc systems, etc.), allowing the storage and processing of huge amounts of data. Similarly, it is possible to develop complex models and algorithms to gather knowledge from such huge databases.

A classical approach to getting information from data in retail and department stores is through market basket analysis (MBA), frequent item set discovery and clustering techniques such as K-means (Hartigan & Wong, 1979), SOM (Kohonen, 1990). The main idea behind this is to discover purchasing patterns from transactional data. However, when we used these techniques to process real supermarket chain data, the results obtained were of very poor quality. For example, with K-means techniques only one cluster grouped 93% of transactions and the 7% remaining is not meaningful. Therefore, poor quality information was generated disabling decisions such as finding customers profiles, discount offers generation, supermarket products layout, etc. Thus, we developed a novel approach to perform MBA based on graph mining techniques; specifically using overlap communities, that allows to generate highly related products to each other within the community. We benchmarked our method using several traditional approaches applied over millions of transactional data. The results of our evaluation show that our approach out–performs the traditional methods.

2. Definitions and related work

This work is focused on generating frequent item sets of products based on transactional data generated by a retail chain. The main idea is to obtain sets of meaningful products so we can generate customer profiles, product layout and recommendations from related products.

In the following sections we will explain the datasets over which we apply our methods; the classical approach and the state-of-art techniques based on graph mining over transactional data.

2.1. Data

We have data from two retail chains in Chile. One is a wholesale supermarket oriented to supply products to grocery store owners, hereafter, referred to as *Retail A*. The second is member of one of the biggest retail holdings in Chile called *Retail B*.

* Corresponding author.
E-mail addresses: ividela@dcc.uchile.cl (I.F. Videla-Cavieres), srios@dii.uchile.cl (S.A. Ríos).
URLs: http://www.ceine.cl (I.F. Videla-Cavieres), http://www.ceine.cl (S.A. Ríos).

0957-4174/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.eswa.2013.08.088

Our data was gathered within a period of thirty months, around 238 million transactions, approximately 160 thousand clients and over 11 thousand SKUs[1] in the case of *Retail A* chain. For *Retail B*, the gathered period was two months, with 128 million transactions, almost 2 million customers and 31 thousand different SKU.

2.2. Transactional data

We have a set of products and transactions. Products are defined formally as $P = \{p_1, p_2, \ldots, p_n\}$ where each p_i represents a specific SKU available. Indeed $|P| = number\ of\ distinct\ SKUs$. A transaction T is defined according to (Agrawal & Srikant, 1994) as a set of items (products in this case) purchased in the same buying opportunity, such that $T \subseteq P$.

In our datasets, products are organized in a three hierarchical level structure. Each level belongs to its predecessor based on an ad–hoc developed taxonomy by each retailer. Fig. 1 shows a subset of one of our taxonomy and Table 1 shows an example of product information with its hierarchy. *Retail A* has 23 product families, 150 lines of products and 415 sublines of products. *Retail B* has 50 product families, 287 lines and 1032 sublines of products.

This big amount of data are stored in a column oriented database because a classical relational database has a very low performance and the response time for every query took several hours or days, which is not acceptable.

Each transaction is identified by a unique number. An example of a transaction set is shown in Table 2 where we see that 925 is a transaction composed of three products: P1, P2 and P4. These products were bought by customer 10021 on the date May 7th, 2009. Suppose SKU of P1 is 13231. On Table 1, that would mean that the product is a Milk named "The Happy Cow" which belongs to Dairy Family, to Yogurt & Milk Line and to Liquid Milk Sub-line. On the other hand, transaction 926 has a *customer id* equal to -1 which means that retail does not have that customer registered or that the customer does not want to give their identifier.

Table 2 presents the set of data available and how that information is stored. Another way to store that information is by the one expressed in Table 3 which is a matrix whose rows are vectors of purchases. Each vector is composed by transactions and the set of products available. The first column stored the transactional id and in the following columns stored a number 1 or 0 which represents whether the product was purchased or not in that particular transaction.

2.3. Market basket analysis

This is one of the most applied techniques over transactional data. It is part of the vast family of Data Mining Techniques. The purpose of market basket analysis is to get a customer to spend more money based on two different principles: the first one is *Up-Selling*, which consists in buying a large quantity of the same product, or adding new features or warranties. The second way is *Cross-Selling*, which consists in adding more products from different categories.

The main purpose to discover frequent item sets. Also known as the discovery of if-then rules called *Association rules* (Agrawal et al., 1993; Agrawal & Srikant, 1994). The form of an association rule is $I \rightarrow j$ where I is a set of items (products) and j is a particular item. The process consist of finding sets of products (items) presents in a large number of transactions (basket).

2.4. Frequent item sets

Frequent item sets are formally defined, according to (Rajaraman, 2012), as follows:

[1] SKU: Stock Keeping Unit.

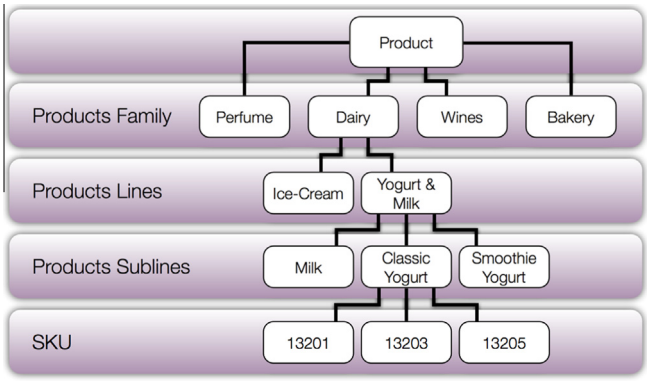

Fig. 1. Hierarchy of products.

Table 1
Products characterization available.

SKU	Product name	Product family	Product line	Product sub-line
13231	Milk "The Happy Cow"	Dairy	Yogurt & Milk	Milk
13201	Yogurt "Fancy Yogurt"	Dairy	Yogurt & Milk	Classic Yogurt
13245	Yogurt "Smoothiest"	Dairy	Yogurt & Milk	Smoothie Yogurt

Table 2
Example of a transaction set.

Transaction ID	Date	SKU	Customer ID	Quantity	Price	Total Price
925	05-07-2009	P1	10021	1	350	350
925	05-07-2009	P2	10021	3	500	1500
925	05-07-2009	P4	10021	2	500	1000
926	05-07-2009	P3	−1	4	600	2400
926	05-07-2009	P4	−1	9	500	4500
927	05-07-2009	P1	1308	4	350	1400
927	05-07-2009	P3	1308	7	600	4200

Table 3
Example of a transaction set as a vector of purchase.

Transaction ID	P1	P2	P3	P4
925	1	1	0	1
926	1	0	1	1
927	1	0	1	0

Let I be a set of items. Define *support* s as the number of transactions for which I is a subset. We will say I is frequent if its support s is bigger than a certain s' called support threshold.

Another important definition related to *association rules* is the *confidence* of a rule $I \rightarrow j$ which is defined as $\frac{support(I \cup j)}{support(I)}$. (In other words the fraction of the baskets with all of I that also contain j). Confidence can be interpreted as the probability of finding the right–hand–side of the rule (in this case j) under the condition that these transactions also contain the left–hand–side of the rule (in this case I).

We performed an experiment using this technique and found very poor results, obtaining a lot of meaningless rules or rules that apply only to a certain group of customers. For example, one of the rules found in the data of *Retail A* is *coke* → *rum*, with a high support and confidence despite the small values obtained in general (less than 0.15% of the transactions). This rule can be seen as a very good rule, but it is an expected rule because in Chile, a common drink named *ron-cola* is made from a base of mixed rum and coke.

Table 4
Top 18 families per cluster after execute K-means algorithm with $K = 7$.

Clusters	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5	Cluster 6	Cluster 7
% Transactions involved	5,1	10,8	46,8	5,5	11,4	13,4	7,0
Families	Yoghurt	Soft Drinks	Cigarettes	French Fries	Milk	Yoghurt	Long Noodles
	Milk	Nectars	Milk	Cookies	Yoghurt	Frozen Desserts	Sugar
	Sugar	Milk	Cheese	Snacks	Frozen Desserts	Milk	Rice
	Toilet Paper	Cookies	Cookies	Milk	Nectars	Cookies	Tomato Sauce
	Vegetable Oil	Yoghurt	Nectars	Souffle	Cheese	Nectars	Vegetable Oil
	Tomato Sauce	Mineral Water	Toilet Paper	Yoghurt	Margarine	Cheese	Short Noodles
	Margarine	Beers	Margarine	Nectars	Butter	Milk	Toilet Paper
	Juice (Powder)	Cheese	Sugar	Soft Drinks	Cookies	Margarine	Yoghurt
	Cheese	Toilet Paper	Vegetable Oil	Biscuits	Soft Drinks	Sausage	Milk
	Short Noodles	Sausage	Sausage	Soft Candy	Juice (Powder)	Toilet Paper	Tea
	Rice	Wine	Juice (Powder)	Pretzels	Milk	Juice (Powder)	Margarine
	Long Noodles	French Fries	Beers	Cookies	Toilet Paper	Biscuits	Salt
	Cookies	Juice (Powder)	Frozen Desserts	Frozen Desserts	Sausage	Pretzels	Juice (Powder)
	Bleach	Sugar	Wine	Hard Candies	Sugar	Soft Drinks	Sausage
	Frozen Desserts	Vegetable Oil	Paper Towels	Chocolate	Sausage	Butter	Bleach
	Nectars	Biscuits	Flour	Juice (Powder)	Vegetable Oil	Sugar	Detergent
	Mayonnaise	Margarine	Chocolate	Cheese	Bleach	Cereals	Cookies
	Tea	Pretzels	Pretzels	Toilet Paper	Delicacy	Sausage	Mayonnaise

This rule also only applies to liquor store owners, not to grocery store owners. In some contexts this can be useful, but for the customer characterization that we are developing, it is not useful.

2.5. Benchmark experiments

We applied K-means and SOM techniques to our data set, obtaining results that were far from being useful. From K-means we basically obtained the same cluster independent of the number of clusters K that we required. Table 4 shows the top 18 categories found by K-means, each column displayed is sorted by the membership of each product family to the respective cluster.

It is clear from Table 4 that clusters are very similar and showing no real difference between them. Also, analysts said that these clusters are meaningless to them, resulting in no new information from this clusterization. As a manner to depict this fact: Cluster 5 and Cluster 6 are very similar. (In fact the top three families are the same in both clusters.).

In the case of SOM we obtained basically meaningless information. These facts motivated us to propose a new methodology to generate clusters of products related between them as we describe in Section 3. This methodology tries to find a meaningful item set with an approach based on product network with overlapping community detection.

3. Proposed methodology

In previous the section we present that classic approach do not give meaningful results. This motivated us to propose a new methodology based on product networks and overlapping communities detection as a frequent item set detection algorithm. In this section we will show how are generated these products network and will show our proposal of a *Temporally Transactional Weighted Products Networks*. Later we will present what is understood as community and their relation with frequent item set.

We present in this section our approach that is a novel way of generating frequent itemsets through community discovery. From now on we will refer to frequent itemsets as *community*.

3.1. Product network and graph construction

A network is defined as a set of elements interconnected between each other. A common way to represent a network is using a *graph*. A *graph* is a manner to specify relationships between sets of items. Formally, a graph consists of a set of objects, called *nodes* with some pairs of them connected by links called *edges*.

A *product network* is defined as a network where nodes represent products and edges represent relationships between a pair of them. We have to define what kind of relationship is represented by an edge. In this case, an edge between two products represents that both products are present in the same ticket from the same buyer opportunity.

We use a network representation based on transactional data shown in Section 2.2. In this subsection we will show how we build our product network.

Kim, Kim, and Chen (2012) show a way to build a bipartite customer product network, that links users with products, though we believe that this approach is limited because it only generates links between customers that have been already identified and lose valuable information that can be obtained by using the whole set of transactions available. We prefer to generate a network of products in the same way as (Raeder & Chawla, 2009), based only on transactions where each product is linked to others because they appear in the same ticket from the same buyer opportunity, this kind of network is named *co–purchased product network*. We then apply a temporary set of filters to check quality and stability of the communities found. We named this generated networks *Temporally Transactional Weighted Product Network*.

The building process of the *temporally transactional weighted product network* is divided in three equally important phases. The first one is to build a set of temporal (daily, weekly, monthly, quarterly, semesterly, yearly) information with the same structure as the one exhibited in Table 3. Once we have this information, we start to build our transactional product bipartite network where each transaction is linked to the products that are purchased in that particular transaction (as in Fig. 2(a)). Here, the set of transactions T and products P represents the two disjointed sets required to build a bipartite graph. Finally, we move from that bipartite network to the co-purchased product network as shown in Fig. 2(b).

After this processing we obtained a product–to–product weighted network. These networks can be represented by an adjacency matrix showing the weight between each pair of products. The weight is the number of tickets, in which a couple of products are present simultaneously. In Table 5 we show this representation.

Similarly to (Raeder & Chawla, 2009) we found the same problems, such as very dense products networks with highly

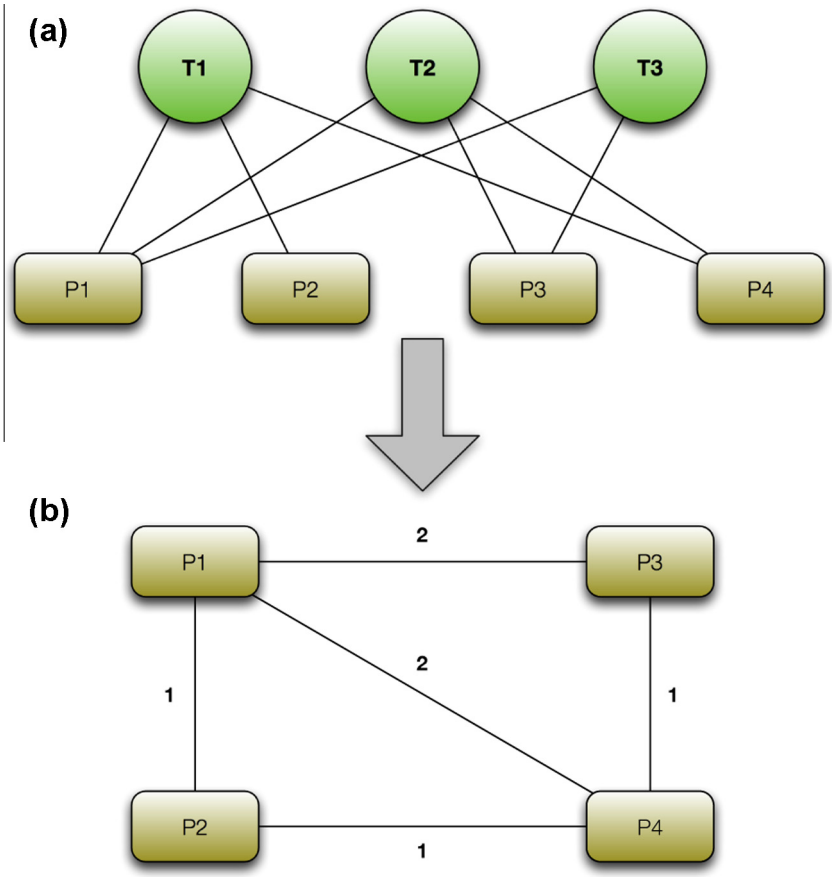

Fig. 2. From bipartite transaction products network (a) to product-to-product undirected weighted network (b).

Table 5
Adjacency matrix representing a product-to-product weighted network.

	P1	P2	P3	P4
P1	–	1	2	2
P2	1	–	0	1
P3	2	0	–	1
P4	2	1	1	–

degree nodes. Most of the time these edges make no sense and represent spurious associations. For example, a month of data has 485,136 transactions, the adjacency matrix of products network obtained has 5,359 products with 5,362,906 edges. Like the adjacency matrix is symmetrical we only consider one half of it. The most heavy weighted edge is 31,224. From those edges, 4,235,093 have a weight lower than 10. This should not be considered because it represents only a sporadic and not frequent relation between products.

3.2. Community detection

Before we explain the overlap community detection process, we have to explain what is understood by *Community Detection* in graphs. It is the process of trying to find a group of strongly connected nodes. According to Fortunato (2010) the first problem is to look for a quantitative definition of community, because usually, the definition depends on the specific system or application developed. Basically the main idea is that there should be more edges *inside* the community than edges linking vertices of the community with the rest of the graph. Moreover, most of the cases, communities are algorithmically defined without a precise a priori definition. Mathematically the problem of finding communities inside a graph is describe as follow:

Given a graph (or network) $G = \{V, E\}$, where V is a set of n nodes and E is a set of m edges, a series of disjoint subgraphs $K = \{K_1, \ldots, K_j\}$ are generated. The number j of subgraphs to find is not known previously and is determined by the algorithm, based on the maximization of a function $f(K)$. So typically, this function is *modularity* (Newman & Girvan, 2004), and is defined over a set of communities as:

$$modularity = \sum_j (e_{jj} - a_j^2) \tag{1}$$

where e_{ij} is the fraction of edges that join vertices in community j to the rest of the vertices of community j. a_j is the fraction of edge endpoints that lie in community j. The idea behind modularity is simple, a subgraph is a community if the number of links among nodes in the subgraph is higher than expected if links were randomly placed.

3.3. Overlapping community detection

This is an extension of classic community detection. The main difference is that subgraphs are not necessarily disjoint which means that a node may belong to more than one subgraph. The set of subgraphs found is called a *cover* $C = \{c_1, c_2, \ldots, c_k\}$ (Lancichinetti & Fortunato, 2009). Each node i is associated with a community according a belonging factor $[a_{i1}, a_{i2}, \ldots, a_{ik}]$ (Nepusz, Petróczi, Négyessy, & Bazsó, 2008) where each a_{ic} is a measure of the strength of the association between node i and cluster c.

In this context (of overlapping communities), it is possible to distinguish two forms of overlapping. *Crisp* or *non-fuzzy*

overlapping where each node belongs to one or more communities with equal strength: a network vertex either belongs to a community or it does not. On the other hand, in *fuzzy* overlapping, each vertex may belong to more than one community but the strength of its membership to each community may vary. In both cases, without loss of generality, this constraint can be assumed:

$$0 \leqslant a_{ik} \leqslant 1, \ \forall i \in V, \ \forall k \in C$$

and

$$\sum_{k=1}^{|C|} a_{ik} = 1, \ \forall i \in V \tag{2}$$

where $|C|$ is the number of clusters in cover C.

3.4. Algorithms for overlapping community detection

An important number of algorithms have been developed to discover overlapped communities. These algorithms vary in effectiveness and performance depending on the type of network.

Xie, Kelley, and Szymanski (2013) present a state of the art in overlapping community detection. They present algorithms and categorized each one into five different classes based on the way in which communities are identified. Founded on this work (Xie et al., 2013) we decided to use the two algorithms with best performance –including quality of results and execution time–. These are COPRA (Gregory, 2010) and SLPA (GANXiS nowadays) (Xie, Szymanski, & Liu, 2011). Both are based on the label propagation algorithm (Raghavan, Albert, & Kumara, 2007), in which nodes with the same label form a community. COPRA updates its belonging coefficients by averaging the coefficients from all its neighbors. Otherwise SLPA is a general speaker-listener algorithm based in the process of information propagation. SLPA spreads labels between nodes according to pairwise interaction rules. SLPA provides each node with a memory to store received information in difference to COPRA where a node forgets knowledge gained in the previous iterations.

3.5. Threshold setup methodology

We showed that these product networks presents high degree nodes with spurious edges between them. To remove spurious edges, a threshold s has to be defined, then the graph is fully revised in the search of edges with a weight s' lower than s ($s' \leqslant s$). The edges that match with this criteria are removed. Raeder and Chawla (2009) decided to filter those edges that have a weight lower than 10. Kim et al. (2012) filter the *co–purchased* network by choosing a threshold s equal to the average value of all links.

We found that there is no common criteria to choose a threshold. This makes it highly necessary to remove these spurious edges, in an objective way, because it is clear that a particular number (constant) like 10 or the average value of all links are very particular thresholds that apply to particular instances or certain data. For instance, in our case, 10 is not a good threshold because the network obtained after applying this threshold still contain spurious edges and isolated nodes that does not produce communities of good quality.

We generate this threshold based on a process denominated *top three heavy edges threshold* (tthet) which was used in both retailer data, proving its effectiveness. This approach consists in ranking the edges $E = \{E_1, E_2, \ldots, E_m\}$ based on the weight of these in a descendant order. Then *tthet* is equal to the average of the top three edges.

$$tthet = \frac{E_{max} + E_{2nd\ max} + E_{3rd\ max}}{3} \tag{3}$$

where E_{max} makes reference to the heaviest edge, $E_{2nd\ max}$ and $E_{3rd\ max}$ to the second and third heaviest edges respectively.

3.6. Network filter methodology

In the case of *Retail A* we obtained 1,492 *Temporally Transactional Weighted Product Networks* and in the case of *Retail B* we obtained over 12,000. To each one of these *Temporally Transactional Weighted Product Networks* we computed a threshold using Eq. (3).

If we apply the obtained *tthet* to its corresponding network, only one or two elements would satisfy the minimum edge weight imposed by the threshold. Since tthet allows us to keep the most relevant part of the *Temporally Transactional Weighted Product Network*, making useless the analysis.

This fact prompted us to generate a set of filters using the tthet, that allow gradually incorporating relevant edges and nodes into our analysis. These filters are a proportion of the *top three heavy edges threshold* (proportion is a percentage of the threshold). The percentages are: $Percentage = \{5\%, 10\%, \ldots, 95\%, 100\%\}$; these percentages give 20 filters (or new thresholds), as a result of a dot product between *percentage* and *tthet* resulting in:

$$filters = percentage \otimes tthet$$

equal to:

$$filters = \{0.05 * tthet, 0.1 * tthet, \ldots, 0.95 * tthet, tthet\}$$

It is clear that for each threshold we have a set of twenty filters associated with the same *Temporally Transactional Weighted Product Network*.

We applied immediately these filters to *Temporally Transactional Weighted Product Networks* giving as a result a new set that we denominated as *Filtered Temporally Transactional Weighted Product Networks* composed of a 29,500 filtered networks in *Retail A* and over 200,000 in the case of *Retail B*.

Fig. 3 depicts the number of nodes and edges from one month of transactional data after applying *filters* obtained from *top three heavy edges threshold*. Is clear that when the percentage goes up; the number of nodes and edges go down in a power law figure.

One of the main advantages of choosing a threshold this way is their independence of the underlying data. This means that our threshold and filters are independent of the quantity of nodes and work very well with big networks both in number of nodes or edge weight. It is also objective because it only depends on the data and requires no intervention from the analyst or a thorough understanding of the business, which is desirable, but not a prerequisite. Thus, our methodology can be reproduced by other works, allowing them to compared their results with ours.

3.7. Communities of products

Graph theory is applied in many fields including analysis of social networks (Ríos & Muñoz, 2012), the world wide web (Faloutsos, Faloutsos, & Faloutsos, 1999), epidemiology (Moore & Newman, 2000), scientific collaboration (Lu, Janssen, Milios, Japkowicz, & Zhang, 2006; Redner, 1998).

Fig. 4 depicts a product-to-product network for *Retail A* on a particular day before any filter was applied. As we can see, this network is meaningless because it has many edges with little value (as we explained in Section 3.5).

Now, when a filter is applied, the network shows meaningful zones as we can see in Figs. 5 and 6. These figures were obtained after we applied a filter equal to the 5% and 10% of the *top three heavy edges threshold*.

These zones describes products with a powerful relationship between them. To understand these relationships and giving them

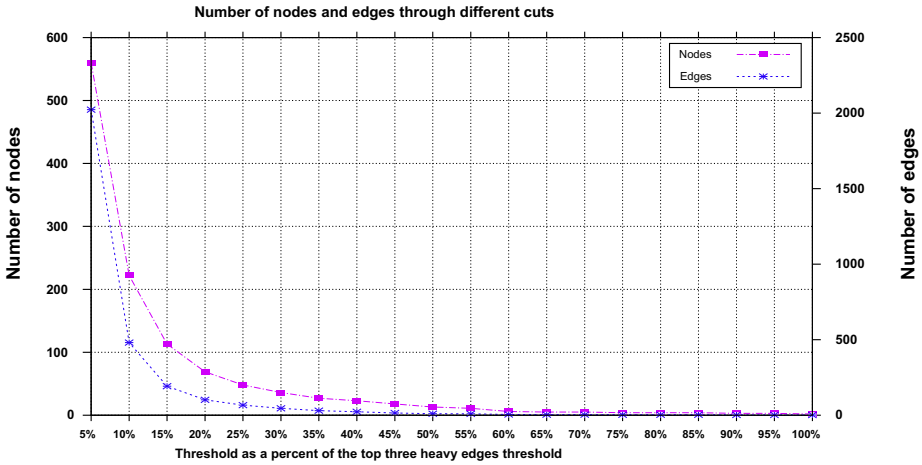

Fig. 3. Graph depicting the number of nodes and edges through different cuts over a month period.

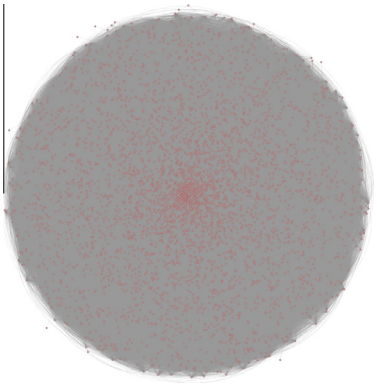

Fig. 4. Visualization of the product-to-product network without filters.

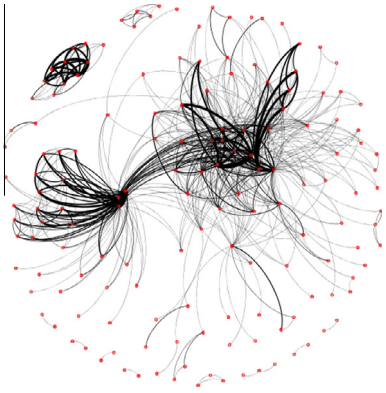

Fig. 6. Product-to-product network with a 10% filter.

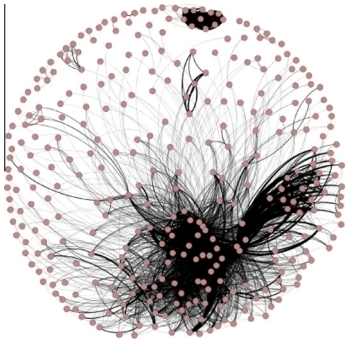

Fig. 5. Product-to-product network with a 5% filter.

a meaningful interpretation is that we applied an overlap community detection process.

After we have generated our set of *Filtered Temporally Transactional Weighted Products Networks* it is time to apply the algorithms for overlapping community detection, described previously in Section 3.4. We apply to each of the *Temporally Transactional Weighted Product Networks* (over 29,000 and 12,000 for *Retail A* and *Retail B* respectively) COPRA and SLPA algorithms.

Both algorithms take a network as an input and generate a file as an output with the communities discovered inside. COPRA generates only one file with the communities and SLPA generates a number of files that are the product of the number of repetitions that a user selects –10 in our case– and a threshold r. Taking values in $[0,1]$ ($r \in [0,1]$ specifically (0.01;0.05;0.1;0.15;...;0.5)) and used as a filter to the number of labels that, a particular node can have checking if the probability of seeing a particular label during the whole process is lower than r. If that occurs, this label is removed from node's memory. When $r \to 1$ the algorithm tends to find disjointed communities. This is explain because only nodes in one community can have a high probability that can overcome the threshold represented by r. After the entire process the nodes are grouped into communities with the same label. If a node has more than one label it is grouped into several communities.

SLPA is executed in a series of runs. A run is an execution of an SLPA algorithm. In each run different values of r ($r \in \{0.01, \ldots, 0.5\}$) are used. Every time the algorithm is executed, a new node is chosen randomly making it necessary to run the SLPA algorithm several times, to avoid that because SLPA started from a '*good*' node the results were improved. We try to find the results that are maintained over time or between executions.

Once the *Filtered Temporally Transactional Weighted Product Networks* are processed by the algorithms we have as result over 3.2 million files in *Retail A* and over 18 million files in *Retail B*.

3.8. Evaluating results

Having 101 files for each *Filtered Temporally Transactional Weighted Product Networks* it is necessary to find a way to discover which file (from 101 available) has the best community representation based on a criteria. As explained in Section 3.2, this criteria is *modularity*.

Table 6
Metadata of a temporally transactional weighted product network for november, 2012.

Network	# of Nodes	# of Edges	Period	Begin date	End date
graph_month_201211	23,615	24,153,638	Monthly	2012-11-01	2012-11-30

To obtain modularities we apply a program to each file with their corresponding *Filtered Temporally Transactional Weighted Product Network* and obtained an equal number of modularity text files. These were parsed and inserted into a column oriented database so we could filter by different criteria such as time window, number of nodes, number of edges and modularity value.

As a manner to depict the process we show in Table 6 the metadata of a Temporally Transactional Weighted Product Network for November, 2012 for a set of supermarket stores from *Retail B*.

In Table 7 the metadata of one of the twenty filtered networks obtained after apply a threshold is shown. The threshold applied is equal to 1,286 –equivalent to the 5 % of the *top three heavy edges threshold*–. Finally, in Table 8, the information of the results gathered from the appliance of SLPA and COPRA algorithms are presented.

We present a subset of the results from SLPA (10 from 101 available) and the result obtained from COPRA in Table 8. This table shows the results for SLPA algorithm after one run.

Table 7
Metadata of a filtered temporally transactional weighted product network for november, 2012.

Network	# of Nodes	# of Edges	Threshold
filtered_graph_month_201211	356	1,133	1,286

Table 8
Results from one iteration of SLPA and COPRA algorithms.

Network	Modularity	# of communities	# of products	# overlaps	r
SLPA_1	0,586	28	358	2	0.01
SLPA_2	0,602	29	360	4	0.05
SLPA_3	0,607	29	358	2	0.10
SLPA_4	0,607	29	358	2	0.15
SLPA_5	0,607	29	358	2	0.20
SLPA_6	0,607	29	358	2	0.25
SLPA_7	0,607	29	358	2	0.30
SLPA_8	0,610	29	357	1	0.35
SLPA_9	0,617	29	356	0	0.40
SLPA_10	0,617	29	356	0	0.45
SLPA_11	0,607	28	356	0	0.50
COPRA_1	0,394	45	361	5	N/A

Table 9
10 largest communities discovered (order by number of products inside) which account for 85% of the products in the network.

Community	# of products	Description
1	242	Grocery
2	15	Soft Drinks & Beers
3	10	Convenience Food
4	6	Juice Powder Brand A
5	6	Juice Powder Brand B
6	6	Liquid Juice Brand C
7	5	Yoghurt Brand D
8	5	Yoghurt Brand E
9	5	Liquid Juice Brand F
10	5	Cookies Brand G

As we can see from Table 8; Both algorithms utilize all products available –356 in this case–. For example, in SLPA_7 the number of products is 358, because 2 are overlapped. The COPRA algorithm discovered more communities than the SLPA algorithm, but modularity obtained from COPRA is worst than SLPA due to the fact that 30 communities from COPRA are singletons. On the other hand, the larger number of overlapped products found by COPRA is explained by the fact that COPRA found similar communities that only differed in one product. For example, for a pair of communities found there is $a = \{269324, 901093, 901095\}$ and $b = \{269324, 901096, 901095\}$, which are almost the same community except, for the middle product. A product identified by SKU equal to 901096 is missing in the case of a and SKU equal to 901093 in the case of b.

We also found a relationship that is present in almost all results obtained from SLPA, that the higher number of overlapped products is found when $r \leqslant 0.2$. This is because when r is small ($r \to 0$), too many nodes can overcome this threshold, as explained previously.

3.9. Discovered communities

Once the algorithms were applied, we had as a result a set of communities of products, where each product is related to each other. In this section we will describe the communities found, in terms of the meaning of this products within the community.

Table 9 depicts the top ten communities ordered by the number of products inside. We also provide a description from the analysts from *Retail B* who study the products involved and gave this description.

It is very important to note that these results changed the opinion of the business analysts, because they believed that people were not *loyal* to a specific brand –in the sense of buying products of the same brand–. However, the results showed that people are loyal to a brand in most cases, the only exception being *Drinks* & *Beers*.

The average number of products inside a community is 7, giving to the analyst a manageable number of products, that can be interpreted and characterized. This is one of the advantages of our work,

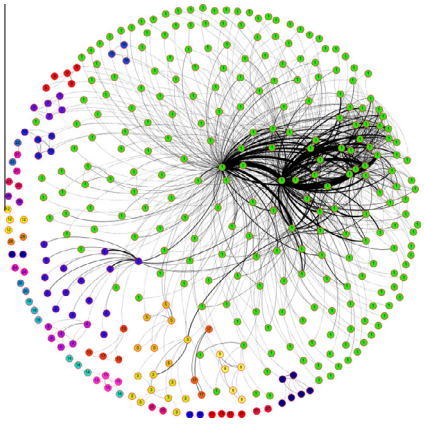

Fig. 7. Visualization of the product-to-product network with each product associated to their corresponding community.

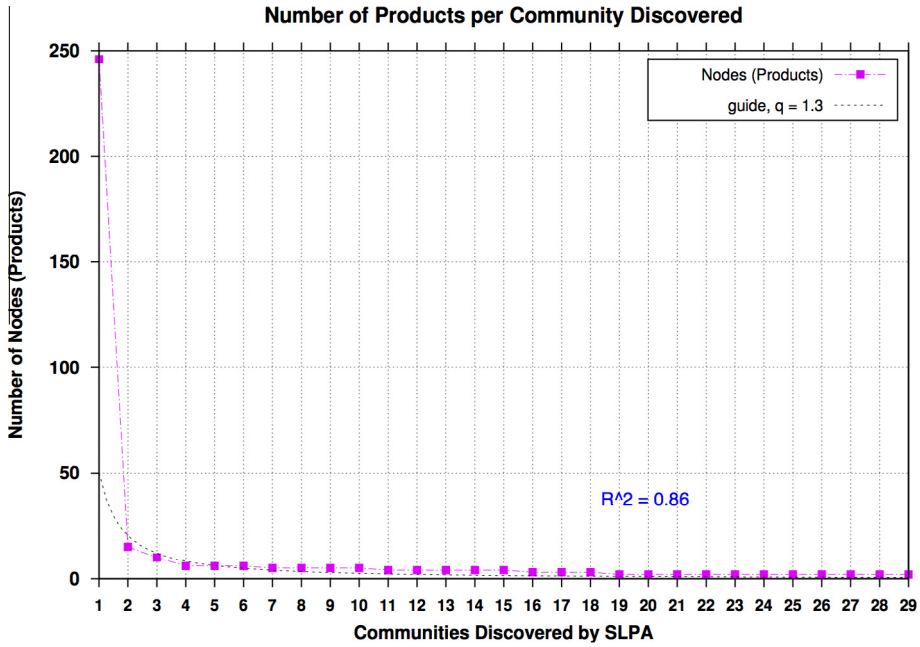

Fig. 8. Number of products per community discovered by SLPA.

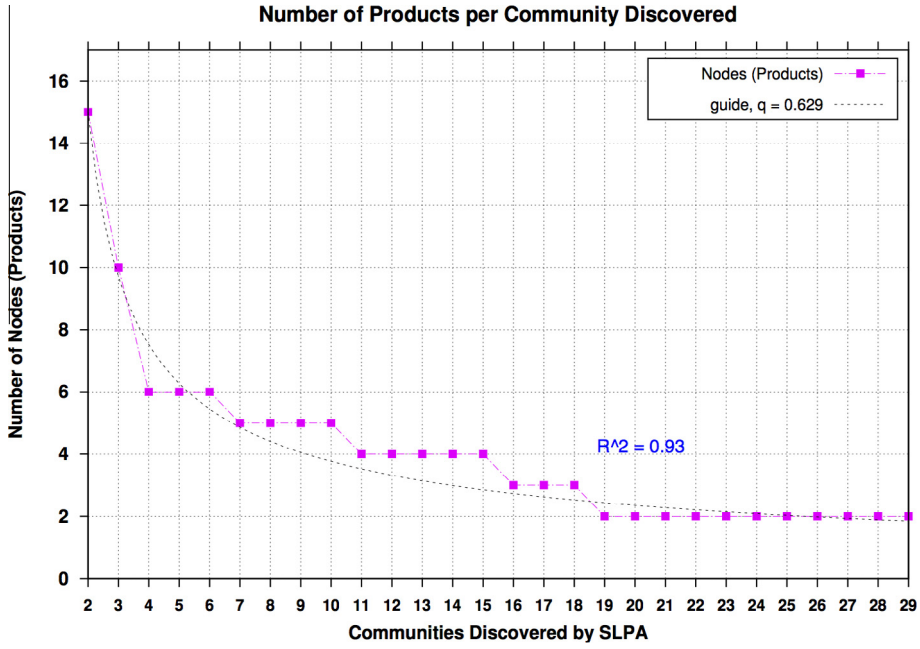

Fig. 9. Zoom to communities discovered by SLPA.

because for a lower quality information from the association rules. A deep and complex analysis must be performed.

We previously present in Section 3.7 how a graph looks after we apply our *top three heavy edges threshold*. Now in Fig. 7 we show how the graph looks with each node colored according their corresponding community. We found a big community of groceries according to an analyst description depicted in green, and the rest of communities in different colors.

We found a property also present in Clauset, Newman, and Moore (2004) and Arenas, Danon, Díaz-Guilera, Gleiser, and Guimerà (2004) that is when a network is partitioned in such a way to maximize modularity, the community sizes q appears to have a power-law form $P(q) \sim q^{-w}$ for some constant w. In this case this constant is $w \simeq 1.3$ considering all the communities available and $w \simeq 0.63$ if we exclude the first community. The number of products in each community are depicted in Fig. 8. The respective power-law function is also plotted.

Fig. 9 is a zoom of the plot presented in Fig. 8. It leaves the first community out of the plot and only shows the 28 communities remaining. The *coefficient of determination* denoted by R^2 is 0.86 considering all the communities and 0.93 if we remove the first community. From both figures and the value of R^2, it is clear that power-law function fits very well with the number of elements found in each community.

4. Conclusion

We have shown a novel approach that uses graph mining techniques to perform market basket analysis and the use of overlapping community detection algorithms as frequent item set discovery technique. Is also presented as a way to extract useful information from millions of product sales transactions.

We introduce the concept of *Temporally Transactional Weighted Product Network* which allows to cover different information needs from retailers and other organizations. These networks can be sliced by a mix of different criteria, such as time window (daily, weekly, monthly, etc.), a particular store, a cluster of stores or the entire *Retail* transactions.

We propose an objective methodology for threshold and filters setup in order to reduce noisy data (independent from the nodes and edges quantity). Firstly, we proposed the *top three heavy edges threshold* method. An unsupervised threshold that only depends on the data available. Secondly, we defined filters which are a proportion of the *top three heavy edges threshold*. Combining both, we ensure that the study can be reproduced and can also be extended to other data sets.

Subsequently, we propose the application of overlapping community detection algorithms as a manner to generate frequent item sets. We performed a benchmark using state–of–the–art algorithms COPRA and SLPA. We also apply state–of–the–art frequent item set algorithms such as K-means, SOM and Apriori. The benchmark was realized in a wholesale supermarket chain and a retail supermarket chain, processing around $238,000,000$ and $128,000,000$ transactions respectively.

We asked the *Retail* business experts to compare our methodology results with traditional market basket analysis algorithms like Apriori, K-Means, and SOM. We discovered that results from traditional techniques were far from being useful, because clusters were formed by huge amounts of mixed products, thus, a segmentation based on these results was meaningless. The main reason was the spars nature of supermarket data and the big size of information involved.

However, with our methodology we were able to produce meaningful and useful frequent item sets. For example, using K-means we obtained mainly 2 representative clusters and one of these concentrate 93% of the products (around 14,000), versus our method, which found –in the same data– 30 clusters (communities) with an average of 7 products. It is clear that 7 products is a manageable number for any analyst. Another example are Apriori algorithm results which gave several rules with very low support and confidence.

Our approach has shown that it can be used as a valid technique to discover frequent item sets present in transactional data. Is important to remark that the information given is very useful for retail, depicting relationships that were not obvious for the analyst of the *retail*. For example, that people is loyal to a particular brand instead of a mix of brands as expected.

In future work, one could try to use these communities of products to generate a fuzzy customer profile. This profile could be generated based on previous purchases mixed with product communities. Amongst other applications, product communities together with customer profiles could be useful for generating personalized recommendations based on customer's historical preferences.

Acknowledgements

The authors would like to acknowledge the support of the Chilean Millennium Institute of Complex Engineering Systems (ICM: P05-004-F FIN. ICM-FIC). We also would like to acknowledge to the Business Intelligence Research Center (CEINE) for their continuous support.

References

Agrawal, R., & Srikant, R. (1994). Fast algorithms for mining association rules. In *Proceedings of 20th international conference on very large data bases, VLDB* (pp. 1–32).

Agrawal, R., Imielinski, T., & Swami, A. (1993). Database mining: A performance perspective. *IEEE Transactions on Knowledge and Data Engineering, 5*(6), 914–925.

Arenas, A., Danon, L., Díaz-Guilera, A., Gleiser, P. M., & Guimerà, R. (2004). Community analysis in social networks. *The European Physical Journal B – Condensed Matter and Complex Systems, 38*(2), 373–380.

Clauset, A., Newman, M., & Moore, C. (2004). Finding community structure in very large networks. *Physical Review E, 70*(6), 066111.

Faloutsos, M., Faloutsos, P., & Faloutsos, C. (1999). On power-law relationships of the internet topology. *ACM SIGCOMM Computer Communication, 29*(4), 251–262.

Fortunato, S. (2010). Community detection in graphs. *Physics Reports, 486*(3), 75–174.

Gregory, S. (2010). Finding overlapping communities in networks by label propagation. *New Journal of Physics, 12*(10), 103018.

Hartigan, J., & Wong, M. (1979). Algorithm AS 136: A K-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics), 28*(1), 100–108.

Kim, H. K., Kim, J. K., & Chen, Q. Y. (2012). A product network analysis for extending the market basket analysis. *Expert Systems with Applications, 39*(8), 7403–7410.

Kohonen, T. (1990). The self-organizing map. *Proceedings of the IEEE, 78*(9), 1464–1480.

Lancichinetti, A., & Fortunato, S. (2009). Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities. *Physical Review E*, 1–9.

Lu, W., Janssen, J., Milios, E., Japkowicz, N., & Zhang, Y. (2006). Node similarity in the citation graph. *Knowledge and Information Systems, 11*(1), 105–129.

Moore, C., & Newman, M. (2000). Epidemics and percolation in small-world networks. *Physical Review E, 61*(5), 5678.

Nepusz, T., Petróczi, A., Négyessy, L., & Bazsó, F. (2008). Fuzzy communities and the concept of bridgeness in complex networks. *Physical Review E, 1*, 1–13.

Newman, M., & Girvan, M. (2004). Finding and evaluating community structure in networks. *Physical Review E, 69*(2), 026113.

Raeder, T., & Chawla, N. V. (2009). Modeling a store's product space as a social network. In *2009 International conference on advances in social network analysis and mining* (pp. 164–169). IEEE.

Raghavan, U., Albert, R., & Kumara, S. (2007). Near linear time algorithm to detect community structures in large-scale networks. *Physical Review E*, 1–12.

Rajaraman, A., & Ullman, J. D. (2012). *Mining of massive datasets*. Cambridge University Press.

Redner, S. (1998). How popular is your paper? An empirical study of the citation distribution. *The European Physical Journal B – Condensed Matter and Complex Systems, 200*(i), 1–4.

Ríos, S., & Muñoz, R. (2012). Dark Web portal overlapping community detection based on topic models. In *Proceedings of the ACM SIGKDD workshop on intelligence and security informatics* (pp. 1–7).

Xie, J., Szymanski, B., & Liu, X. (2011). SLPA: Uncovering overlapping communities in social networks via a speaker-listener interaction dynamic process. In *ICDMW 2011, 11th IEEE international conference on data mining*.

Xie, J., Kelley, S., & Szymanski, B. (2013). Overlapping community detection in networks: The state of the art and comparative study. *ACM Computing Surveys, 45*(4), 1–37.