Product Recommendation Algorithms in the Age of Omnichannel Retailing – An Intuitive Clustering Approach

Jaydeep Balakrishnan[1]
Chun-Hung Cheng[2*]
Kam-Fai Wong[2]
and
Kwan-Ho Woo[2]

1. Haskayne School of Business, University of Calgary, Calgary, Alberta T2N 1N4, Canada
2. Department of Systems Engineering and Engineering Management, The Chinese University ofHong Kong, Shatin, N.T., Hong Kong SAR

*****Corresponding Author

Published in *Computers & Industrial Engineering*, 115, 2018, pp 133-150.

Product Recommendation Algorithms in the Age of Omnichannel Retailing – An Intuitive Clustering Approach

Abstract

In today's omnichannel retailing world, product recommendations have become important in retailer strategy. Using big data to recommend complementary products can help improve customer service and thereby increase profitability. A common implementation for studying buying behaviour of customers uses a 0-1 matrix linking the customers to the products they have purchased in the past. However, this raw matrix does not automatically reveal buying patterns. Further processing of this matrix is necessary to find valuable information. In this work, we adopt an intuitive co-clustering algorithm for locating useful patterns in the matrix. The advantage of duplication of products in the clustering process will be shown. A further advantage of the algorithm from a managerial perspective is that it is intuitive rather than a black box type and thus may increase the chances of it being actually adopted.

Keywords: buying patterns, product recommendation, 0-1 matrix, co-clustering, branch-and-bound, duplication,

Acknowledgement

This research has been partially supported by a grant from Asian Institute of Supply Chain & Logistics (Account No. 8116023) and a GRF grant (Project No. 14615015) from Research Grants Council of Hong Kong SAR, China.

1. Introduction

The emergence of omnichannel retailing and big data has resulted in product recommendations becoming an important issue for consideration in retail. Omnichannel retailing refers to the use of multiple channels such as physical and online to provide goods and services. Brynjolfsson et al. [2013] discuss how retailers can succeed in the age of omnichannel retailing by taking advantages of what physical location as well as online retailing offer. They note that the way to success include exclusive product bundling that might be difficult for competitors to match. Further a recent McKinsey study (MacKenzie et al. [2013]) notes that with omnichannel retailing, retailers can do personalized advertising and promotion via devices such as the increasingly ubiquitous smartphone. Finally the increasing cost of real estate is forcing retailers to reduce the size of physical stores (especially in high rental locations like Hong Kong), thus forcing them to make decisions on how to select the products in physical stores.

A white paper by Oracle (Oracle [2011]) states that personalized product recommendations improve retailers' e-commerce sales and that research and experience with large online retailers confirmed the value of recommendations. At the same time the paper cautions that despite the attractiveness, product recommendation implementation on online sites has challenges such as: the inability to understand shoppers' preferences and present relevant recommendations; a lack of control provided to retailers to manage recommendations; limited reach into and understanding of each product catalogue; an expensive and lengthy implementation process; an inadequate understanding of how recommendations fit into the overall e-commerce and retail strategy. Thus it is clear that whether one is discussing the choice of what products to display side by side (or perhaps even a physical recommendation notice for complementary products by a 'Did you remember to buy…' sign) or a graphic of a recommended

complementary product on the website, the issue of this selection is important. At the same time, the actual implementation of this process of recommendation can be challenging. In this paper we propose a method that will help identify the groups of similar products from a customer perspective so that retailers can use this information to recommend complementary products similar to when online retailers say 'Customers that purchased this products also purchased…'. Naturally all this can be done by a live salesperson too.

Retailers have reams of data in corporate databases. If properly extracted and processed, it provides valuable knowledge for managing the product recommendation process. The process requires an understanding of analytical data mining techniques. These technologies are used to analyse buying patterns of customer's sales data. Then optimization techniques can be used to make complementary product recommendations. Based on these recommendations, salespeople may focus on promoting selected products to a group of customers, or these complementary products may be displayed together in the store display. Alternately as mentioned previously, even if they are not displayed proximally, there could be signs reminding the customers to purchase the complementary product wherever else in the store the product may be located (indeed placing the complementary product in another aisle may provide the opportunity for the retailer to gain impulse purchases from the customer on his/her way to purchase the complementary product). Also, as discussed earlier, this focused approach to marketing improves customer service and increases profitability for the retailer. In addition, such an approach makes effective and efficient use of sales effort as well as physical location. For example, given that many retailers have reduced their floor space, it is important to select the limited groups of products to stock in-store for maximum effect.

In terms of online selling an appropriate recommendation for complementary products ensures the credibility of the site. A situation where the recommendations are not effective is counterproductive. Consider the case where the lack of recommendation of a crucial complementary product results in a customer receiving the online purchase but is not able to enjoy the product or service to the maximum because of the lack of the complementary or alternate product. This is may result in the customer being unhappy with the retailer and posting this dissatisfaction in the product review page on line.

In this work, we plan to use clustering to help us make product recommendations. Clustering refers to a class of data mining methods that assign dis-similar objects to different groups and similar objects to same groups (Anderberg [1973] and Han et al. [2011]). These clustering methods require the problem applications formulated in a specific representation. A common representation is a data matrix with rows and columns. In this work, our matrix connects the customers and the products they have purchased, with the customers being the rows and the products being the columns of the matrix. To cluster this matrix, some techniques (referred to as single-dimension clustering methods) deal with the rows and columns separately while others (referred to as co-clustering methods) handle the rows and columns simultaneously.

Co-clustering methods produce row clusters and column clusters at the same time. They allow the information presented in all the rows and columns of a matrix to be considered in the clustering process. In single-dimension clustering, only the information from either the rows or columns is used for producing column clusters or row clusters, respectively. Further, Long et al. [2005] claimed that co-clustering methods are particularly suitable for handling sparse data. The sparsity of sales data is often a source of poor clustering results (Gong [2009] and Gong [2010]).

In a realistic situation, many customers purchase a small number of a company's products. Hence, co-clustering seems to be a better approach for processing sales data.

The rest of the paper is organized in this way. Section 2 describes the problem. We propose to use a matrix representation for modelling sales data. This representation is then suitable for the use of co-clustering methods. Section 3 discusses co-clustering techniques and applications. Relevant work will be reviewed. In Section 4, we discuss the formulation of the problem and the solution algorithm by Cheng et al. [2011]. Cheng et al's method adopts a branch-and-bound framework and is originally used to solve a data partitioning problem in database design. We modify Cheng et al's method for our purpose. In Cheng et al, their objective is to come up with the best partitions of data. We are more concerned with the managerial implications from their method when it is applied to a business problem. The implementation issues concerning the development of the algorithm and the computational experience are discussed in Appendix. Section 5 shows the effectiveness of the algorithm. Section 6 concludes this paper.

2. Problem Statement

A 0-1 matrix is used to model sales data, with the rows being the customers and the columns being the products. One such matrix is shown in Figure 1. In this matrix, an entry $a_{ij} = 1$ means that customer *i* has purchased product *j*, and it is 0 if this purchase has not happened. So in Figure 1, customer A purchased products 2, 4 and 5 while customer B purchased products 1 and 3. In the bottom of the matrix, we show the total sales for each of the products. The use of this will be discussed later. With the matrix in this format, purchasing patterns are difficult to detect. However, if the rows and columns of the matrix are properly rearranged, noticeable patterns (i.e., co-clusters) will be visible in the resulting matrix (see Figure 2). The rearrangement

of the rows and columns of a matrix to reveal co-clusters in the resulting matrix is referred as co-clustering in this work.

Put Figure 1 here.

Put Figure 2 here.

In Figure 2, customers B and D show the same buying behaviour and customers A and C have a very similar buying pattern. Since customer C did not buy product 5, even though it is in the same co-cluster as customer A, it may be advantageous for a salesperson to recommend and attempt to push product 5 to the customer (since customer A purchased it, it might indicate a complementary product). If the customer still rejects the product, he/she may reveal to the salesperson the aspects of the product that he/she does not like. It could be for example because of the price being too high, or the product lacking the features the customer wants. This may lead to reformulating marketing strategies for the product (e.g., reduction in price) or developing a new product with the desirable features pleasing the customer.

It is important to form compact co-clusters. Otherwise, customers may be approached with many products that they do not like. Pushing too many unwanted products to the customers has two disadvantages. First, it may annoy the customers with repeated sales calls (or annoying messages in online retailing) that do not provide information interesting to the customers. Second, it may also waste the valuable time of the salespersons pushing products to the customers who actually do not want them. Similarly if the issue is related to co-placement physically, placing inappropriate items together is a waste of valuable space. The important advantage of co-clustering is that we identify common customer groups (customers B and D from one group and customers A and C from another group) who can be targeted for similar personalization in marketing while other clusters can be used to perform different personalization. This

understanding will have ripple effects up the supply chain. The existence of groups implies that the production systems can also be designed to optimally produce these co–clusters of the products and the transportation system can be designed to optimally transport these co–clusters of the products. This should lead to economies of scale throughout the supply chain leading to decreased production and transportation costs and increased profitability. At the same time the firm should also experience increased customer service by ensuring that these entire co-clusters are available upon customer demand.

In practice, a matrix with perfectly disjoint (separate) co-clusters is very rare. Figure 3 gives a different but perhaps more realistic example. Assuming for a moment that product 3 did not exist, the matrix displays two clean co-clusters like in the example in Figure 2. Customers E and F would be one co-cluster and customers G and H would be another co-cluster. It will also be clear that in this case, product 5 can be recommended to customer H. So from a promotion decision making perspective, this is a welcome situation.

However in reality we have to deal with product 3. Should product 3 be grouped with the EF co-cluster or with the GH co-cluster? To avoid the analysis issue, we could just ignore product 3. However this may result in loss of the selling information about product 3. To avoid losing information on this product, we adopt a duplication strategy instead in our co-clustering algorithm. Product 3 will be duplicated to allow the formation of the disjoint co-clusters in the resulting matrix. The total sales for product 3 will be split between customers F and G based on their past purchases (recorded in the sales database). With these different co-clusters, product 3 may be recommended to customer E and products 3 and 5 may be presented to customer H.

Put Figure 3 here.

The model discussed so far is more applicable if each product has approximately the same sales. In this case, the total sales of the products need not be considered. So in the problems in Figure 1 and 3, we could assume that the each of products 1 to 5 had one instance of sales or that each of the five products had equal instances of sales (in which case we could still use a 0 or 1 to denote no demand or demand occurrence, as this is the simplest representation).

However this may not be true in practice. For example consider the situation in Figure 2 where the volume of sales for Product 2 and 4 to customer C may be three times as large as that to customer A. In order to include this information in the algorithm that solves the problem, we have an additional row that sums the total sales volume for each product.

Real-life databases containing sales information could be quite large. Thus an effective algorithm is needed to take the problem that may be in the format shown in Figure 1 and convert it into the format shown in Figure 2. So in the paper we suggest an algorithm and test it to see whether it would be feasible to solve the product recommendation problem described in the previous section.

3. The Co-clustering Approach

We modify the co-clustering approach developed by Cheng et al. [2011]. This 2011 algorithm improves a previous algorithm by Cheng et al. [1995], for cellular manufacturing and data partitioning. Since the Cheng [1995] algorithm may fail to produce feasible solutions in some problem instances, Cheng et al. [2011] propose a new branching strategy. Their algorithm provides the following advantages:

1. Their algorithm does not need a pre-specified number of co-clusters in the resulting matrix. This is a useful property as it is often difficult to know the suitable number of co-clusters apriori.
2. Product selling information will not be lost in the co-clustering process as they adopt duplication rather than removal strategy. As discussed earlier, removal strategy may be undesirable as it often results in loss of sales information.
3. As it will be illustrated later in this work, the co-clustering mechanism is very simple. It can be easily implemented and applied. Unlike many existing co-clustering algorithms (reviewed in the following section), the clustering process is intuitive and can be easily demonstrated and explained to the potential users.

3.1 Review of Literature

Direct clustering (or co-clustering) of a data matrix involving cases and variables simultaneously were possibly first discussed by Hartigan [1972]. His heuristic method was used to analyse voting data. Since Hartigan's work, many researchers have developed different co-clustering approaches and applied them to different fields. Since the literature is rich, we briefly discuss various relevant approaches using a structured framework.

There are two main streams of approaches in co-clustering: heuristic-based and optimization-based methods. Heuristic-based methods can be further divided into hierarchical and non-hierarchical types and optimization-based methods include mainly graph-theoretic and cluster identification types.

Hierarchical methods are used in areas such as biological and medical research. Madeira and Oliveira [2004], and Tanay et al. [2006] give a review of these methods. They refer to co-

clustering as bi-clustering. These methods may form an extended hierarchical clustering tree along the rows and columns of a data matrix. They provide very compact co-clusters and do not require a prior number of co-clusters. The users may get very good quality co-clusters by manually choosing a cut in the trees. The number of co-clusters will be determined at the same time as when the cuts are chosen.

Non-hierarchical methods are found for instance in text classification. A double clustering method may be used in which feature clustering is followed by document clustering (Slonim and Tishby [2000]). Feature clustering reduces noise and sparseness so that high quality document clusters can be generated. To further improve a double clustering approach, Yaniv and Souroujon [2001] propose an iterative double clustering. In successive iterations, noisy data are reduced to improve a co-clustering result. In an information-theoretic method, Dhillon et al.'s [2003] approach starts with an arbitrary partition of rows (features) and columns (documents) and obtains an approximation of the underlying distribution. In iterations, it performs row-clustering taking column-clustering information into consideration and vice versa. The algorithm stops when the underlying distribution is almost found. The solution matrix will then reveal natural co-clusters. In a recent work, Seyedhosseini et al. [2015] apply a group-based league championship algorithm to a manufacturing design problem. On the contrary, our work uses an algorithm in design to a product promotion problem in marketing.

Graph-theoretic methods are commonly found in text classification as well. Instead of considering document and word clustering separately, Dhillon [2001] deals with them simultaneously by modeling a bipartite graph between documents and words. Co-clustering is regarded as a bipartite graph partitioning problem. To solve this problem, a spectral co-clustering algorithm, possessing desirable optimality properties, is proposed.

Kusiak and Chow [1987] propose a cluster identification method. This is an optimal method to produce disjoint co-clusters in the input 0-1 matrix provided that these disjoint co-clusters exist in the matrix. However, their algorithm will fail if the input matrix does not have disjoint co-clusters. So for example in Figure 3, since Product 3 overlaps between the EF and GH product groups, the algorithm would not be able to form separate co-clusters. Cheng [1995] embodies cluster identification with a branch-and-bound framework to allow the formation of disjoint co-clusters. He proposes to remove columns which prevent disjoint co-clusters from happening but his removal strategy may produce infeasible solutions in some problem instances. In a follow-up work, Cheng et al. [2011] fix the infeasibility by adopting a duplication strategy.

In this research, we adopt Cheng et al.'s [2011] approach to solve the co-clustering problem for product recommendation. The use of clustering in this problem is not new. Gong [2009], Gong [2010], Zouzias et al. [2012], Vlachos et al. [2014], Datta [2015], and Heckel [2017] are some clustering examples for product recommendation. Gong [2010] addresses the problem of the sparsity in sales data. Zouzias et al. [2012] do not require a prior number of co-clusters. Vlachos et al.'s [2014] process is similar to K-means and hierarchical clustering approaches in the co-clustering process. Datta et al. [2015] deal with the scalability problem of product recommendation. Heckel et al. [2017] handle the problem of producing recommendation by locating co-clusters of customers and products. Compared to the existing approaches, our approach using Cheng et al. [2011] is much simpler and direct. Thus it is more likely to be accepted by users as a more intuitive method. Unlike many co-clustering approaches, we use duplication strategy to clustering the data matrix. The advantage of using duplication strategy is that information on columns will be not lost in the clustering process. Also because of duplication strategy, the block-diagonal structure will be automatically formed.

4. Co-clustering with Duplication Strategy

We adopt Cheng et al.'s algorithm [2011] to solve the co-clustering problem. In this section, we will discuss the formulation of the problem and summarize the major processing components of the algorithm. For the details of the implementation of the algorithm, the readers may refer to Cheng et al. [2011].

4.1. Problem Formulation

This formulation must permit duplication of products to allow disjoint co-clusters in the resulting matrix. As we discussed earlier, removal strategy in Cheng [1995] may cause loss of sales information on some products. Hence, a duplication strategy is a preferred option. Similar to Cheng et al. [2011], we have a quality constraint for co-clusters formed. In this constraint, a measure must be used to indicate the quality of a co-cluster. This measure used has a value between 0 and 1.

To measure the quality of a co-cluster formed, Cheng et al. [2011] developed a cohesion measure. With a 0-1 co-cluster S that has m rows and n columns, we have $S = [a_{ij}]$ where $i = 1, ..., m$ and $j = 1,..., n$. We also let R be the set of all the row indices in S, and C be the set of all the column indices in S. The cohesion measure of co-cluster S is $\delta_S = | \{ a_{ij} = 1, i \in R \text{ and } j \in C \} | / (|R|*|C|)$. In the other words, the cohesion measure is the ratio between the number of "1" entries in a co-cluster and the size of the co-cluster. For example in Figure 2 for the Customer B and D, Product 1 and 3 co-cluster, the cohesion measure $\delta_S = 4/(2*2) = 1$ means that the customers and products are a perfect match. Interesting though, from a product recommendation perspective it implies that there is no opportunity in this cluster to suggest products. In contrast for the Customer A and C, Product 2, 4 and 6 co-cluster, the cohesion measure $\delta_S = 5/(2*3) =$

0.8333. In this case since $\delta_S \leq 1$, therefore the customers and products are not a perfect match. So this gives us the opportunity to recommend products to customers as explained earlier.

So this problem is different from many other applications of clustering where the objective is to make the clusters as dense as possible. A perfect co-cluster may not be a good result from a product recommendation perspective. Consider a 0-1 matrix that can be grouped into ten clusters with the average δ_S being 0.9. Suppose in one case this average of 0.9 is derived from five clusters where $\delta_S = 0.8$ and five clusters where $\delta_S = 1.0$. This means that in the five co-clusters with $\delta_S = 1.0$, no recommendation is possible while in the other five recommendations may be possible. Now consider the same ten clusters with the average δ_S of 0.9, but with each of the ten individual co-clusters also having $\delta_S = 0.9$. In this case we may be able to make product recommendations for ten groups rather than five as in the previous case.

We formulate the co-clustering problem allowing duplication of products as follows:

P1: Minimize the total sales of the products being duplicated when producing disjoint co-clusters in a 0-1 matrix subject to the following constraints:

C1: a customer with no purchase in a co-cluster is not allowed.

C2: a product with no customer purchase in a co-cluster is not allowed.

C3: The cohesion measure of a co-cluster cannot be less than a threshold, δ.

In Problem P1, the objective is to minimize the total sales of the products being duplicated. The products and their total sales will be both considered in forming co-clusters. For a removal strategy, the objective could be to minimize the total sales of the products being removed. Constraint C3 set a lower bound for the threshold to ensure the acceptable level of the quality of a co-cluster. A user may specify the value of the threshold. This value ranges from 0.1 to 0.9. Constraints C1 and C2 (preventing a co-cluster having a customer with no purchase or a

product with no customer from being formed) must be enforced throughout the co-clustering process.

4.2. Solution Method

Similar to Cheng [1995] and Cheng et al. [2011], we use a branch-and-bound framework. A node is taken from the enumeration tree. This node, just like other notes in the tree, has a 0-1 matrix that might already have one or more co-cluster. The root node has the input matrix. The input matrix is taken as a single co-cluster. The algorithm uses cluster identification developed by Kusiak and Chow [1987] (the details of which will be discussed later in this section) to form co-clusters. If all the disjoint co-clusters satisfying all the constraints are found, the current solution in this node will be compared to the incumbent solution. A new incumbent solution will be found if the current solution is better in terms of the value of the objective function. If a co-cluster (violating some of the constraints C1 to C3) in the matrix cannot be divided into smaller disjoint co-clusters, then the algorithm will find and duplicate a product to decompose the co-cluster. The computation will terminate when there is no node in the tree left unprocessed or unfathomed.

4.2.1 Branch-and Bound Framework

The tree structure is constructed using a stack. The advantage is that the push-and-pop mechanism of a stack enforces a depth-first-search (DFS) discipline. Cheng [1995] provides the following implementation:

Step 0. (Initialization): Initialize the root node containing the 0-1 matrix $[a_{ij}]$. Set the upper bound $Z_U = \infty$ for the total sales of the products being duplicated. If this algorithm terminates with $Z_U = \infty$, it implies that a solution satisfying all the constraints could not be found.

Step 1. (Branching): Based on the depth-first search strategy, select an active node (not fathomed). The depth-first search strategy can be implemented by the push and pop mechanisms of a stack structure. Apply the cluster identification algorithm to a co-cluster of the node that does not satisfy all the constraints. When the co-cluster of the node cannot be decomposed, apply the duplication strategy (discussed later) to the co-cluster.

Step 2. (Bounding): For each new node, find a lower bound, Z_L on the value of the objective function.

Step 3. (Fathoming): Exclude a new node from further consideration if:

Test 1: $Z_L \geq Z_U$ (in this case, this new node will never lead to a child node that is better the incumbent solution),

Test 2: Any corresponding co-cluster violates constraint C1 or C2,

Test 3: All corresponding co-clusters satisfy the three constraints. If $Z_L < Z_U$, store this solution as the new incumbent solution, set $Z_U = Z_L$, and reapply Test 1 to all remaining unfathomed nodes created previously.

Step 4. (Stopping rule): Stop when there are no unfathomed nodes remaining; the current incumbent solution is final. Otherwise, go to Step 1.

4.2.2. Cluster identification

In Cheng [1995], Kusiak and Chow's cluster identification [1987] is used to process a co-cluster in a matrix associated with a node taken from the enumeration tree. However, his implementation of the cluster identification algorithm is not efficient as it requires O(*min(n,m)* ×

$n \times m$) for a regular $m \times n$ matrix. In Cheng et al. [2011], they propose a bitset implementation. Since a bitset is simpler and it may be processed by linear-time bitwise operations, the complexity is determined by the number of the rows (i.e., the customers) only, irrespective of the number of the columns (i.e., the products).

For the convenience of the readers, we reproduce the implementation by Kusiak and Chow [1987] shown as follows:

Step 0. Set iteration number k = 1.

Step 1. Select any row i of the 0-1 matrix $[a_{ij}]^k$ and draw horizontal line h_i through it ($[a_{ij}]^k$ is read as matrix $[a_{ij}]$ at iteration k.).

Step 2. For each entry of "1" crossed by the horizontal line h_i, draw a vertical line v_j.

Step 3. For each entry of "1" crossed once by a vertical line v_j, draw a horizontal line h_k.

Step 4. Repeat Steps 2 and 3 until there are no more crossed-once entries of "1" in $[a_{ij}]^k$. All crossed-twice entries of "1" in $[a_{ij}]^k$ form a co-cluster.

Step 5. Transform the 0-1 matrix $[a_{ij}]^k$ into $[a_{ij}]^{k+1}$ by removing rows and columns corresponding to all the horizontal and vertical lines drawn in Steps 1 through 4. The removed rows and columns form a cluster

Step 6. If the 0-1 matrix $[a_{ij}]^{k+1}$ contains no row and column, Stop; otherwise set k=k+1 and go to Step 1.

Figures 4 through 6 perform the cluster identification algorithm on the matrix in Figure 1. In this example, we start with the first row of the matrix. Actually, it does not matter in which row the first horizontal line is drawn. The final result will be the same. A co-cluster is formed when one cannot draw any more horizontal and vertical lines. The matrix will be reduced by

eliminating the rows and columns belonging to the newly formed co-cluster. The cluster identification algorithm will then be performed on the reduced matrix. As a result, the co-cluster solution like in Figure 2 will be produced. Kusiak and Chow [1987] show that the cluster identification algorithm can always find disjoint co-clusters in a matrix if these co-clusters do exist in the matrix.

4.2.3. Duplication Strategy

In many cases a co-cluster in a matrix cannot be further decomposed due to blocking products. However, it is rather difficult to pinpoint these products. Cheng [1995] introduces the use of void measure to estimate the likelihood of a column (i.e., a product) being a blocking factor. Once a product with the highest void measure is identified, the algorithm then duplicates the product to make the decomposition possible. The total sales for the product chosen to be duplicated will be split among the customers who have purchased it according to the actual purchases by these customers recorded in the sales database. In our approach, we use the rule by Cheng [1995]. To illustrate this rule, Cheng [1995] define

- a_{ij} = entries in the i^{th} row and j^{th} column of a (m x n) matrix M
- Q_j = { $j' \neq j$: $a_{ij'} = 1$ for some i = 1, 2, ..., m such that $a_{ij} = 1$}
- $C_j = Q_j \cup \{ j \}$
- R_j = { i : $a_{ij} = 1$ and (for some $j' \in C_j$: $a_{ij'} = 1$)}.

Then they develop the two-step rule shown as follows:

Step 1 (Evaluation of void measure):

Calculate the void measure of product j, V_j, for every product j in M. The void measure for product j is defined as:
$V_j = | \{ a_{ij'} = 0, i \in R_j$ and $j' \in C_j \} |$.

Step 2 (Duplication):

Sort the products by the value of void measures in descending order. Products will be selected for possible duplication according to descending order of the value of void measure. For each product, split it into a set of products in which each has a unique single entry "1" derived from the original product being considered. The total sales of the product chosen for duplication will be split among all the customers who have purchased it based on their actual purchases.

Take the simple matrix in Figure 3 as an example. The matrix is not clustered originally. The duplication is applied. Product 3 will eventually be found to be duplicated to make the input matrix clustered into two co-clusters. The total sales for product 3 are split between customers F and G according to the actual purchases by the customers recorded in the sales database.

4.2.4. Clusters Merging Process

To avoid forming small co-clusters, a merging mechanism is used to merge co-clusters into a larger one without increasing the value of the objective function and violating any of the constraints. This processing is useful as it reduces the number of co-clusters in a clustering solution.

5. Effectiveness of the algorithm

Cheng et al. [2011] show that the computational complexity of this co-clustering algorithm is $O(k \times (n + m) \times m)$, where n is the number of rows, m is the number of columns, and k is a value representing the number of nodes to be examined by the branching rule. Further they also show that the space complexity of the memory requirements for the input matrix, the stack structure, the branch-and-bound scheme are $O(m \times n)$, $O(n^2)$, $O(m \times n^3)$, respectively.

Consider the enumeration tree in Figure 7 as an example. The input matrix is shown as the root node of the search tree. The total sales for all the products are only shown at the root

node. We assume that the lowest threshold for a cluster, δ, is 0.7. The problem matrix cannot be clustered into smaller co-clusters and hence some products must be duplicated. We find that product 1 has the highest value of the void measure. Product 1 is being duplicated first in the search tree. Cluster identification is then performed on the matrix with the duplication of product 1. Although co-clusters are found, one of the co-clusters does not fulfil the cohesion measure threshold. Further duplication continues. At this time, product 2 is found with the highest void measure and is duplicated. Cluster identification is again performed. Since all the co-clusters formed satisfy the constraints, the set of co-clusters found is set as the incumbent solution and the upper bound is updated. The search then backtracks to other sibling nodes at Level 2. Since these nodes have the lower bound $Z_L = 8$, they are excluded from further consideration. The search backtracks to Level 1 of the search tree and product 2 is duplicated. The duplication of product 2 does not create clusters. A further duplication is required. However, the further duplication will only result in a co-cluster with its $Z_L = Z_U$. Hence, further processing along this path is not necessary. The search continues until there is no unfathomed node in the search tree.

Put Figure 7 here.

Some co-clusters in the solution formed are fragmented. Co-clusters with product {1} and with products {1", 2'} could be grouped together to form a larger co-cluster (see Figure 8) in a solution.

Put Figure 8 here.

To show the application of our algorithm, we consider another larger problem with the threshold being 0.6. In Figure 9, the 0-1 matrix contains 11 customers and 22 products. The initial matrix does not reveal any noticeable structure.

Put Figure 9 here.

The solution matrix is shown in Figure 10. Products 4, 5, 6, 8, 9, 11, 12, 14, 17, 18, and 19 are duplicated. Some products are duplicated twice to form four co-clusters in the resulting matrix.

Put Figure 10 here.

6. Conclusion

In this work, we applied a modified co-clustering algorithm to the product recommendation problem. In this age of omnichannel retailing, product recommendations whether through salespersons, product physical placement in a store or through the internet on portable devices have become very important. This paper dealt with using an effective process to determine product recommendations. This involves clustering products and customers into groups. A 0-1 matrix can was used to represent different clustering problems. Typically when the matrix is constructed, it will not display any patterns. A co-clustering algorithm was used to identify patterns in the matrix. Our co-clustering algorithm uses Cheng et al.'s [2011] approach. Based on the co-clustering process, we identified groups of the customers who have similar buying patterns and recommended products to customers who may be interested in purchasing them. Illustrative examples were used to show the co-clustering processing of the algorithm. The complexity was

analysed to show that the process is indeed effective in providing solutions to the product recommendation problem.

Single-dimension clustering and co-clustering may be used in product recommendation application. Co-clustering allows all the information presented in the rows and columns of a matrix to be used in the clustering process. Single-dimension clustering can only use row or column information in column or row processing, respectively. In addition, co-clustering is a better method to handle sparse matrix data that presents in many applications including product recommendation.

Unlike other co-clustering methods, Cheng et al. [2011] adopt duplication strategy in the clustering process. This strategy is unique. Duplication of products makes the input matrix decomposable and reveals the underlying patterns in the matrix data. Interpretable managerial insights can be derived from visually studying the decomposed matrix and can be easily explained to practitioners who are unfamiliar to clustering.

References

1. Anderberg, M.R. (1973), *Cluster Analysis for Applications*, New York, Academic Press.

2. Brynjolfsson, E., Yu, J. H. and Rahman, M.S. (2013), "Competing in the Age of Omnichannel Retailing", *Sloan Management Review*, Summer.

3. Cheng, C. H. (1995), "A Branch-and-bound Clustering Algorithm", *IEEE Transactions on Systems., Man, and Cybernetics*, Vol. SMC-25, No. 5, pp. 895-898.

4. Cheng, C.H., Wong, K.F. and Woo, K.H. (2011), "An improved branch and bound algorithm for vertical partitioning", *International Transactions in Operational Research*, Vol. 18, No. 2, pp. 231-255.

5. Datta, S., Das, J., Gupta, P., and Majumder, S. (2015), "SCARS: a scalable context-aware recommendation system", *Proceedings of the 3rd International Conference on Computer, Communication, Control and Information Technology*, pp. 1-6.

6. Dhillon, I.S. (2001), "Co-clustering documents and words using bipartite spectral graph partitioning", in *Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 269-274.

7. Dhillon, I.S., Mallela, S. and Modha, D.S. (2003), "Information-theoretic co-clustering", in *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 89-98.

8. Gong, S.J. (2009), "Employing user attribute and item attribute to enhance the collaborative filtering recommendation", *Journal of Software*, Vol. 4, No. 8, pp. 883-890.

9. Gong, S.J. (2010), "A collaborative filtering recommendation algorithm based on user clustering and item clustering", *Journal of Software*, Vol. 5, No. 7, pp. 745-752.

10. Han, JW, Kamper, M., and Pei, J (2011), *Data Mining: Concepts and Techniques*, 3rd edition, Massachusetts, Morgan Kaufmann.

11. Hartigan, J.A. (1972), "Direct clustering of a data matrix", *Journal of American Statistical Association*, Vol. 67, No. 337, pp. 123-129.

12. Heckel, R., Vlachos, M., Parnell, T., and Duenner, C. (2017), "Scalable and interpretable product recommendations via overlapping co-clustering", *Proceedings of IEEE 33rd International Conference on Data Engineering*, pp. 1033-1044.

13. Long, D, Zhang, Z.M., and Yu, P. (2005) "Co-clustering by block value decomposition", *Proceedings of the 11th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 635-640.

14. Kusiak, A. and Chow, W.S. (1987), "An algorithm for cluster identification", *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. SMC-17, No. 4, pp. 696-699.

15. Madeira, S. and Oliveira, A.L. (2004), "Biclustering algorithms for biological data analysis: a survey", *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, Vol. 1, No. 1, pp. 24-45.

16. MacKenzie, I., Meyer, C. and Noble, S. (2013), "How retailers can keep up with consumers", available at http://www.mckinsey.com/insights/consumer_and_retail/how_retailers_can_keep_up_with_consumers, (accessed at January 20, 2016).

17. Oracle Corporation (2011), "An Oracle white paper next-generation product recommendations", available at http://www.oracle.com/us/products/applications/commerce/atg/next-generation-product-333319.pdf , (accessed at February 21, 2016).

18. Seyedhosseini, S.M., Badkoobehi, H. and Noktendan, A. (2015), "Machine-part cell formation problem using a group based league championship algorithm", *Journal of Promotion Management*, Vol. 21, No. 1, pp. 55-63.

19. Slonim, N. and Tishby, N. (2000), "Document clustering using word clusters via the information bottleneck method", in *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 208-215.

20. Yaniv, R. E. and Souroujon, O. (2001), "Iterative double clustering for unsupervised and semisupervised learning", *Lecture Notes in Computer Science*, Vol. 2167, pp. 121-132.

21. Tanay, A., Sharan, R. and Shamir, R. (2006), "Biclustering algorithm: a survey", in S. Aluru (ed.), *Handbook of Computational Molecular Biology*, Chapman & Hall/CRC.

22. Vlachos, M., Fusco, F., Mavroforakis, C., Kyrillidis, A. and Vassiliadis, V.G. (2014), "Improving co-cluster quality with application to product recommendation", in *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pp. 679-688.

23. Zouzias, A., Vlachos, M. and Freris, N.M. (2012), "Unsupervised sparse matrix co-clustering for marketing and sales intelligence", *Lecture Notes in Computer Science*, Vol. 7031, pp. 591-603.

		Product				
		1	2	3	4	5
Customer						
A			1		1	1
B		1		1		
C			1		1	
D		1		1		
Total Sales		5	6	3	4	7

Figure 1 A 0-1 matrix

		Product				
		1	3	2	4	5
Customer						
B		1	1			
D		1	1			
A				1	1	1
C				1	1	
Total Sales		5	3	6	4	7

Figure 2 A re-arranged 0-1 matrix

		Product				
		1	2	3	4	5
Customer						
E		1	1			
F		1	1	1		
G				1	1	1
H					1	
Total Sales		5	6	3	7	5

Figure 3 A more realistic 0-1 matrix

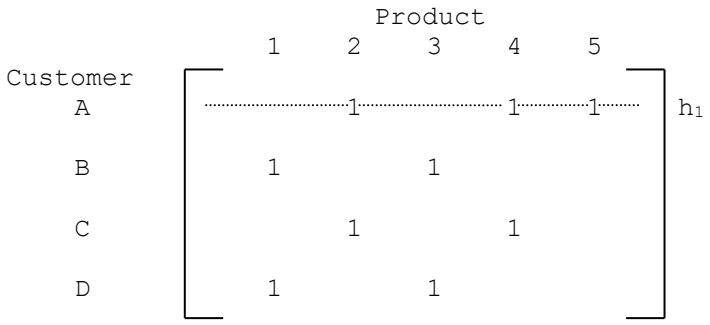

Figure 4 A horizontal line h_1 is drawn

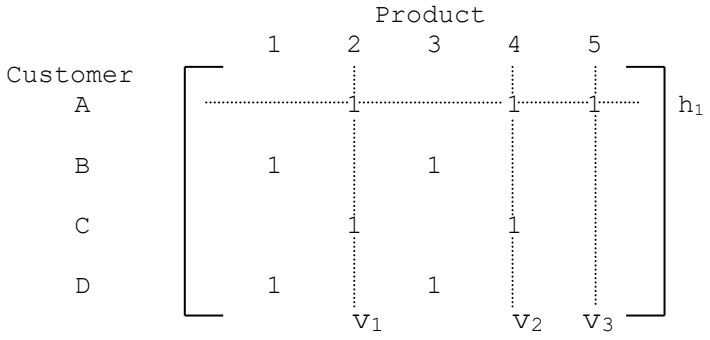

Figure 5 Two vertical lines (v_1 and v_2) are drawn

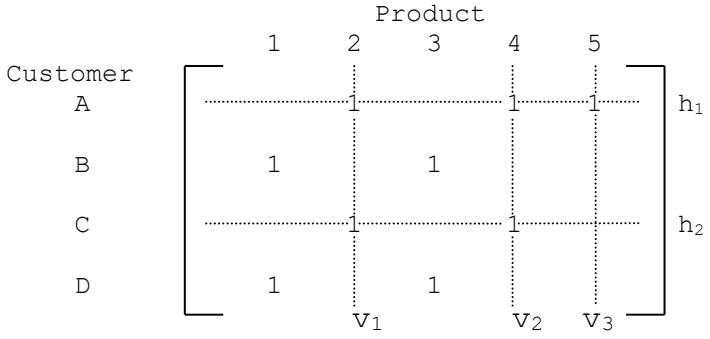

Figure 6 Another horizontal line is drawn

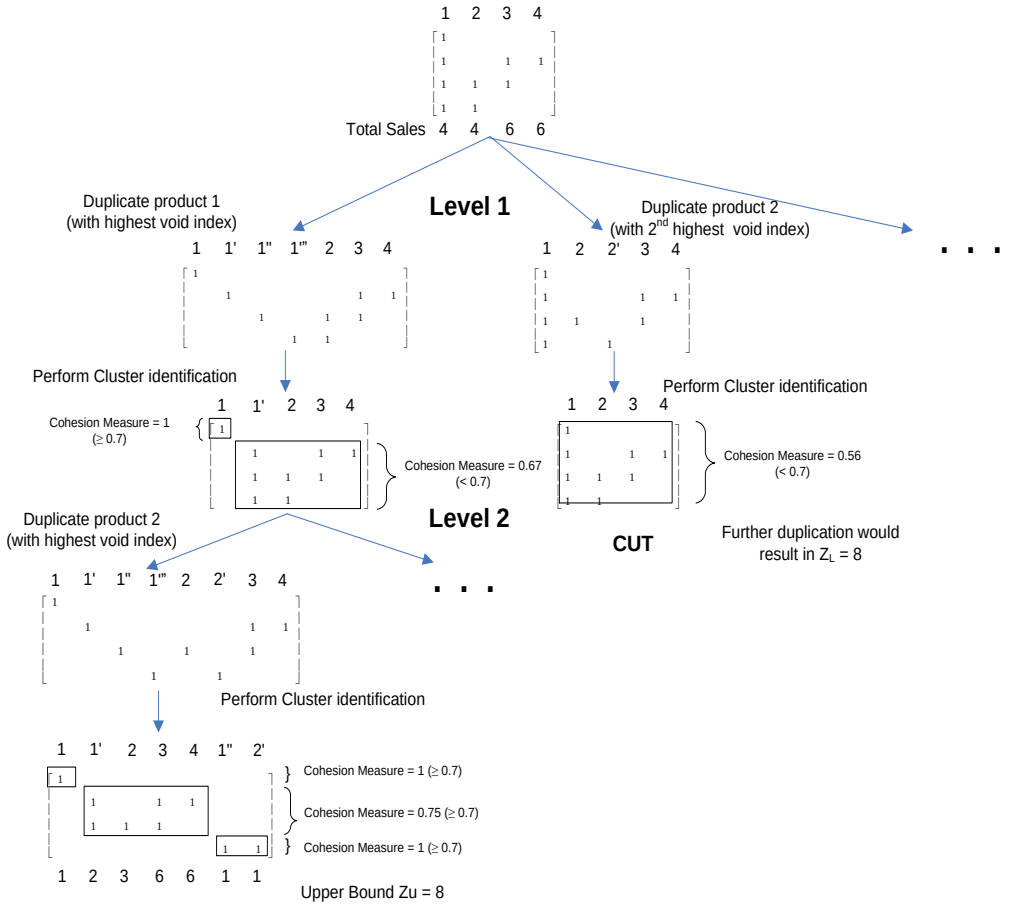

Figure 7 Search tree for illustrating the branch-and-bound approach

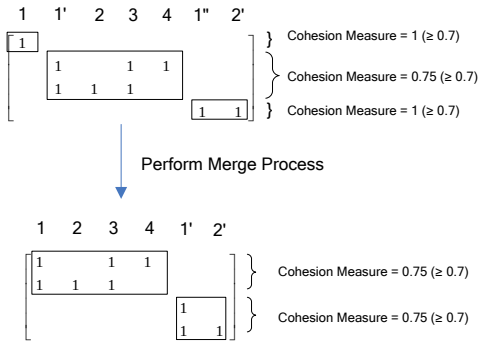

Figure 8 Merge the co-clusters

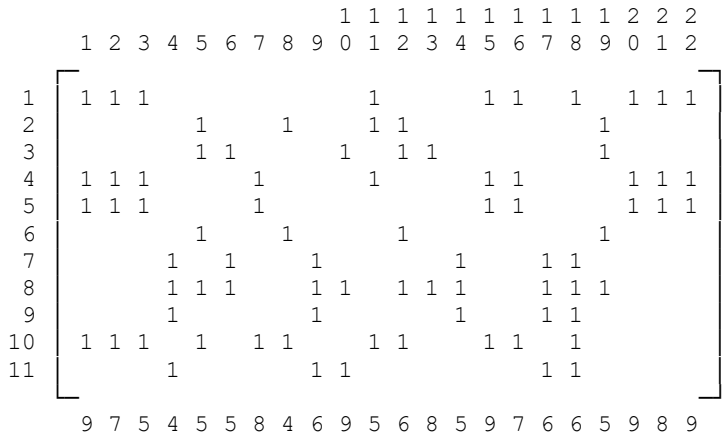

Figure 9 Another Matrix

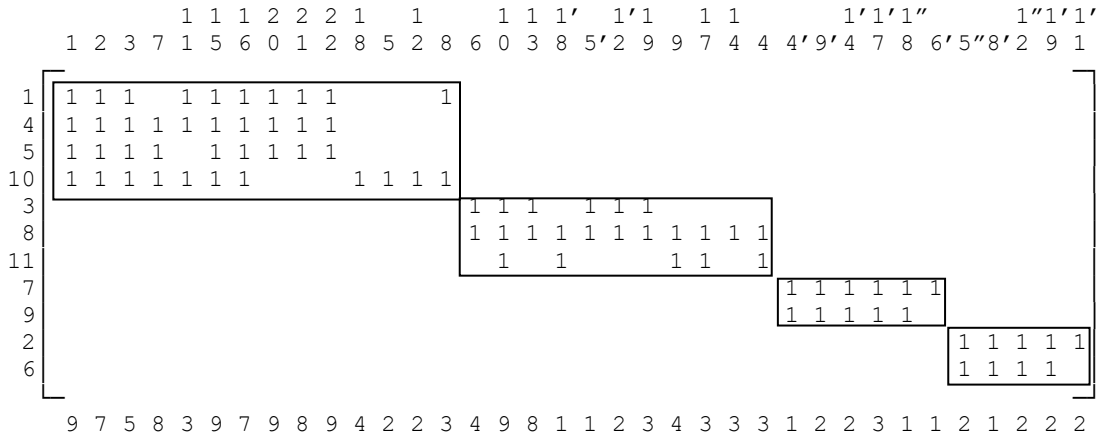

Figure 10 A solution to the matrix in Figure 9

APPENDIX

In this Appendix, we will discuss the implementation issues by covering major functions in the co-clustering algorithm. In addition, we will also present the computational experience of this algorithm. This Appendix provides a summary of this algorithm. Interested readers may refer to Cheng et al. [2011] for a more detailed description.

Co-clustering main function

This function uses a stack to implement the branch-and-bound enumeration tree. The push-and-pop mechanism of the stack ensures a depth-first-search (DFS) strategy. Cheng et al. [2011] provide the following implementation:

Main-Function: Co-clustering
Input:

- Problem matrix M_0: a (m x n) binary matrix represented by a list of row bitsets.
- Threshold δ: the cohesion measure threshold

Output:

- The list of submatrices $List_{final}$: The list of submatrices clustered from Matrix M_0 with the objective function (Z) being minimized

Notation:

- The list of submatrices $List_{in}$: The list of submatrices in an incumbent solution
- The bitset of flag $BF_{redundant}$: The bitset of flag indicating which attribute has been duplicated
- Value Z_L: The lower bound on the objective function (Z) and value Z_U: The upper bound on Z

Algorithm:

1. // Phase 1: Finding inter-submatrix columns
2. // Initialization
3. if (Cohesion Measure of $M_0 \geq \delta$) return to the calling function
4. $Z_U = \infty$
5. $Node_0$ = matrix M_0 with $Z_L = 0$ and $BF_{redundant}$ = null;
6. stack_of_node.push($Node_0$); // Construct the root of enumeration tree
7. while (stack_of_node is not empty) {
8. $Node_{cur}$ = stack_of_node.pop(); // Get the node in DFS tree
9. M_{cur} = matrix in $Node_{cur}$; // Get the corresponding matrix from $Node_{cur}$
10. $Z_{L,\,cur} = Z_L$ in $Node_{cur}$; // Get the corresponding Z_L from $Node_{cur}$
11. perform function CLUSTER_IDENTIFICATION; // Node examination
12. if (($Node_{cur}$ is feasible) and ($Z_{L,\,cur} < Z_U$)) {
13. // Set the list of submatrices from the current node as an incumbent solution
14. L_{in} = list of matrices derived from $Node_{cur}$
15. $Z_U = Z_{L,\,cur}$
16. } else {
17. // Bounding
18. if ($Z_{L,\,cur} \geq Z_U$) {
19. next; // examine other node in the stack by going back to Line 8
20. }
21. // Branching
22. create $List_{cur}$; // Store the set of derived matrices
23. create $BF_{redundant,\,local}$ // Store the flag indicating the redundancy info. of the duplicated attribute.
24. $BF_{redundant,\,local} = BF_{redundant}$ in $Node_{cur}$
25. For each attribute α which can be duplicated in the matrix M_{cur} {
26. // i.e. $\forall\ \alpha \in \{\ a \mid BF_{redundant}[a] = True \wedge a \in \{1,\ldots,n\}\ \}$
27. create new matrix M' by duplicating attribute α in matrix M;
28. set $BF_{redundant,\,local}[\alpha] = True$;
29. Node' = matrix M' with $Z_L' = Z_{L,\,cur} + 1$ and $BF_{redundant} = BF_{redundant,\,local}$;

30. add Node' to $List_{cur}$;
31. } // Duplicate attributes
32. Sort $List_{cur}$ by void measure in descending order;
33. stack_of_node.push ($List_{cur}$);
34. }
35. }
36. // Set the last found incumbent solution as the best feasible solution
37. $List_{final} = List_{in}$;
38. // Phase 2: Merging the set of clusters for alleviating the fragmentation problem
39. perform function MERGE with $List_{final}$

A node of the tree contains one or more submatrices. The root note has the initial matrix and the upper bound is set (Line 4-5). In Line 7-35, a node in a tree is popped from the top of the stack and its submatrices will be used for cluster identification. If cluster identification produces submatrices satisfying all the constraints, a feasible node is obtained. When the new feasible node is better than the current incumbent, the set of the associated submatrices in this node will become the new incumbent and the upper bound will be updated (Line 12-15). However, if the lower bound of the current node is not lower than the upper bound, the node will be pruned (i.e., no new child nodes will be generated from the current node) (Line 16-20). Pruning is critical to the performance of the algorithm as it is useful in cutting down the size of the search space. If both cases are not true, a branching rule (Line 21-33) is applied. A set of new nodes are derived by duplicating the column in a submatrix of the current node. All the child nodes newly generated by the branching rule are then pushed back to the stack.

Cluster identification function

Kusiak and Chow's cluster identification [1987] is one of the main functions in the co-clustering algorithm. When a matrix is separable, this algorithm will always find it. Cheng et al. [2011] provides the following implementation:

Function: CLUSTER_IDENTIFICATION
Input:
- Matrix M: a (m x n) binary matrix represented by a list of row bitsets.
- Threshold δ: the cohesion measure threshold

Output:
- Boolean isFeasible: **true** only if the cohesion measures of all the sub-matrices identified in the matrix M are larger than or equal to the threshold δ
- List of submatrices $List_{sub}$: List of sub-matrices clustered from Matrix M

Notation:
- Matrix M_p: The partial (full) matrix for cluster identification

Algorithm:
1. // Initialization
2. assign $M_p = M$;
3. // Identify all the feasible clusters
4. while (M_P is not empty) {
5. (M_1, M_2) = BINARY_SPLIT(M_P);
6. if (isDecomposable) {
7. if (evaluate_cohesion_measure_of (M_1) ≥ δ) {
8. add M_1 to $List_{sub}$;
9. assign $M_p = M_2$;
10. } else {
11. isFeasible = false;
12. return;

```
13.         }
14.     } else {
15.         if (Cohesion Measure of Mp ≥ δ) {
16.             add Mp to Listsub;
17.             assign Mp = empty;
18.         } else {
19.             isFeasible = false;
20.             return;
21.         }
22.     }
23. }
```

CLUSTER_IDENTIFICATION function decomposes the input matrix M into a set of sub-matrices (i.e., clusters) without violating the cohesion measure threshold δ. Once the matrix M_P is initialized, it will be split by the BINARY_SPLIT function (Line 5). If M_p is separable, it will turn into two submatrices (i.e., M_1 and M_2). M_1 may be a candidate as part of a solution (Line 7-13) while M_2 is the remaining submatrix waiting for further processing (Line 9). When the cohesion measure of M_1 is smaller than the threshold δ, the function will exit the while-loop and return the infeasibility status to the calling function (Line 10-13). The same case applies to a non-decomposable M_p (Line 15-20). BINARY_SPLIT function attempts to split M_p into M_1 and M_2 (Line 5). Non-separable status will be returned to CLUSTER_IDENTIFICATION function if the input matrix is not separable; otherwise, two submatrices will be returned.

Merge function

MERGE function is used to combine submatrices that have been over-split. The implementation of this function in Cheng et al. [2011] is shown as follows:

Function: MERGE
Input:
- δ, the minimum cohesion measure criterion; C_{in}, set of clusters from the clustering approach

Output:
- C_{out}, set of clusters as the resultant compact solution

Notation:
- M_i, i th the sub-matrix(cluster) from the cluster set C_{in}; $|C_{in}|$, the number of clusters in the set C_{in}

Algorithm:

```
1.  Cout ← Cin; set merge_pair to a empty set;
2.  {
3.      δmax ← 0.0;
4.      for Mi ∈ {Cout,i  : i ∈ (1,|Cout|) }{
5.          for Mj ∈ {Cout,j  : j ∈ (i+1,|Cout|) }{
6.              Mmax ← merge(Mi, Mj);
7.              δcur ← evaluate_cohesion_measure_of (Mmax);
8.              if (δcur ≥ δmax) {
9.                  max ← δcur;
10.                 merge_pair ← (Mi, Mj);
11.             }
12.         }
13.     }
14.     merge clusters in merge_pair in Cout;
15. } do while (merge_pair is not empty)
16. return Cout;
```

This function evaluates a pair of submatrices for possible merging (Line 6). Two submatrices may be merged together if the value of the cohesion measure for two submatrices combined at least equals to the threshold (δ). Line 8-10 ensures that two submatrices are merged to generate a new submatrice with the highest value of the cohension measure.

Computational experience

This co-clustering algorithm was implemented in C++ by Cheng et al. [2011]. It has been used to solve ten matrices of different sizes. The maximum running time is limited to 24 hours. When the maximum time is reached, the execution of the algorithm will be terminated. The most recent incombent solution will be taken as the best solution for the problem. Cheng et al. [2011] varied the value of the threshold from 0.1 to 0.9 and examined the impact on the size of the search tree. Tables A1 and A2 show the total number of nodes and the number of nodes required to get the best solution, respectively.

		Cohesion Measure								
Problem	**Size**	**0.1**	**0.2**	**0.3**	**0.4**	**0.5**	**0.6**	**0.7**	**0.8**	**0.9**
1	7 X 11	1	1	43	78	471	901	1,024	1,024	1,024
2	20 X 20	1	36	13,503	109,403	170,117	250,697	261,295	262,144	262,144
3	23 X 20	1	1	71,823	912,615	1.047m	1.048m	1.048m	1.048m	1.048m
4	10 X 22	1	1	64	1,813	1,813	1,813	3.594m	4.194m	4.194m
5	11 X 22	1	1	1	1,813	1,813	1.744m	4.158m	4.192m	4.192m
6	20 X 25	1	1	18,487	932,032	943,759	4.888m	33.539m	33.554m	33.554m
7	12 X 30	1	1	2,975	3,057	3,057	5.658m	10.971m	66.187m	66.187m
8	20 X 50	1	1	145	24,724	16.168m	16.168m	16.168m	82.398m*	86.070m*
9	25 X 35	1	35	194	690	1.677m	7.056m	84.605m*	127.470m*	127.171m*
10	30 X 40	1	5,726	254,220	4.540m	49.678m*	69.016m*	90.504m*	90.672m*	90.549m*

m stands for million times

* indicates that the branch-and-bound process cannot be finished within 24 hours, and that only intermediate result is available

Table A1 Total number of nodes in the search tree during the branch-and-bound process

		Cohesion Measure								
Problem	**Size**	**0.1**	**0.2**	**0.3**	**0.4**	**0.5**	**0.6**	**0.7**	**0.8**	**0.9**
1	7 X 11	1	1	30	20	127	128	11	11	11
2	20 X 20	1	31	932	4,198	41,529	3,006	634	19	19
3	23 X 20	1	1	13,069	4,477	1,026	20	21	21	21
4	10 X 22	1	1	46	5	5	5	547	23	23
5	11 X 22	1	1	1	5	5	75	63	124	124
6	20 X 25	1	1	3,271	229,138	240,865	1.299m	1,700	25	26
7	12 X 30	1	1	5	91	91	343	1,784	241,196	241,196
8	20 X 50	1	1	51	5,085	46,054	46,054	46,054	25.689m*	50*
9	25 X 35	1	2	131	68	1,409	1,264	249,460*	64.691m*	35*
10	30 X 40	1	238	12,239	26,572	137,176*	898,677*	130,358*	33*	33*

m stands for million times

* indicates that the branch-and-bound process cannot be finished within 24 hours, and that only intermediate result is available

Table A2 Total number of nodes have been traversed when the best clustering solution is found

As we observe from Tables A1 and A2, only a small number of nodes as opposed to the number of total nodes have been examined to find the best solution. In addition, the total number of nodes is much smaller than n^n, where n is the number of columns in the problem matrix. It is clear that the algorithm is effective in pruning.

Advantage of Cheng et al. [2011]

Exiting clustering algorithms do not use a duplication strategy in the clustering process. Hence, the objective function and constraints used in Cheng et al. [2011] are very different. This makes direct comparison very difficult.

To make product recommedation possible, we duplicate all the objects (i.e., the products) in every cluster that may possibly hold the objects in a resulting matrix. Let's consider a matrix in Figure A1. A typical solution is shown in Figure A2. If we duplicate product 1 in two other clusters, we will obtain the matrix in Figure A3. This is easily achieved in Cheng et al. [2011], as the duplication strategy is considered in the clustering process.

	Product								
	1	2	3	4	5	6	7	8	9
Customer									
a	1				1	1	1		
b	1				1	1	1		
c	1	1		1					
d		1	1						
e	1							1	1
f								1	1

Figure A1 A simple matrix

	Product								
	1	5	6	7	8	9	2	3	4
Customer									
a	1	1	1	1					
b	1	1	1	1					
f					1	1			
e	1				1	1			
d							1	1	
c	1							1	1

Figure A2 A typical solution

	Product										
	1	5	6	7	1′	8	9	1″	2	3	4
Customer											
a	1	1	1	1							
b	1	1	1	1							
f						1	1				
e					1	1	1				
d									1	1	
c								1		1	1

Figure A3 A solution with duplication