A two level co-clustering algorithm for very large data sets

Bartcus Marius, Boullé Marc, Clérot Fabrice

Orange Labs
prenom.nom@orange.com

Abstract. Co-clustering is a data mining technique that aims at identifying the underlying structure between the rows and the columns of a data matrix in the form of homogeneous blocks. It has many real world applications, however many current co-clustering algorithms are not suited on large data sets. One of the successfully used approach to co-cluster large data sets is the MODL co-clustering method that optimizes a criterion based on a regularized likelihood. However, difficulties are encountered with huge data sets. In this paper, we present a new two-level co-clustering algorithm, given the MODL criterion allowing to efficiently deal with very large data sets that does not fit in memory. Our experiments, on both simulated and real world data, show that the proposed approach dramatically reduces the computation time without significantly decreasing the quality of the co-clustering solution.

1 Introduction

Co-clustering (Hartigan, 1972), also named block clustering (Govaert and Nadif, 2008) or two-mode clustering (Mechelen et al., 2004) is a data mining technique. It aims at identifying the underlying structure between the rows and the columns of a data matrix in the form of homogeneous blocks. Whereas, the principle of standard clustering is to group similar individuals (observations) with respect to a set of features, the task of co-clustering is to simultaneously group similar individuals with respect to variables and similar variables with respect to observations, thus extracting the correspondence structure between the objects and features. Another advantage of co-clustering over standard clustering techniques is its matrix reduction capacity, where a large data table can be reduced into a significantly smaller one yet having the same structure as the original matrix. Indeed, this technique finds its use in many applications like in telecommunications (Guigourès et al., 2015), text mining (Dhillon et al., 2003; Li and Abe, 1998), graph mining (Guigourès et al., 2015), etc.

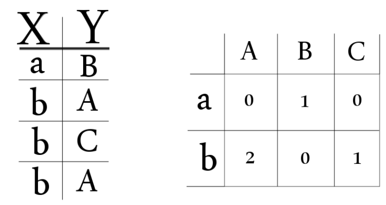
Several co-clustering approaches have been proposed in the literature (Bock, 1979; Dhillon et al., 2003; Govaert and Nadif, 2008). These methods differ mainly according to the type of analyzed data (categorical or numerical), the underlying hypothesis, the extraction method and the expected results. Several families of approaches have then been proposed to perform co-clustering. Govaert and Nadif (2013, 2008) investigated probabilistic models with use of latent variables in mixture models. Difficulties arise on initialization, large number of parameters to estimate and computational efficiency, therefore large data are hard to manage. Indeed, few methods able to co-cluster large data have been proposed in literature. For instance, Papadimitriou and Sun (2008) developed a tool, named DisCo implementing a distributed data pre-processing and co-clustering using Hadoop and a map-reduce implementation. DisCo can scale well and efficiently analyze extremely large data sets, however, it needs a large distributed infrastructure. Another co-clustering method, that exploits probabilistic models for two or more variables of any type (numerical or categorical) is based on the MODL approach (Boullé, 2011). The main advantages of MODL co-clustering is that it is user-parameter-free and benefits from algorithms with sub-quadratic time complexity w.r.t. the number of instances, allowing to deal with large data sets. According to the advantages previously mentioned we focus on the MODL co-clustering approach.

Indeed, the MODL co-clustering can deal with large data sets reaching up to millions of instances and tens of thousands of values per variable, with a sub-quadratic time complexity. However, it can hardly be used with very large data, with up to billions of instances and variables having millions of values. For example, this limit is reached in the case of the analysis of Call Delay Record at a country scale, when the studied granularity goes from antenna level (application to network dimensioning) to individual customers (marketing application with identification of fine-grained communities and customer experience personalization). In this paper, we focus on extending the co-clustering optimization algorithms to these large data, given the MODL co-clustering criterion. Despite the fact that MODL can deal with numerous numerical or categorical and even mixed variables, in this paper we investigate the case of two categorical variables.

This paper is organized as follows. First, for self-containment reasons, Section 2 recalls the principles of the co-clustering method using the MODL criterion (Boullé, 2011) that estimates the joint distribution between two categorical variables. Next, Section 3 introduces the proposed two level algorithm for large data co-clustering. Section 4, gives experimental results and evaluates the proposed approach on simulated and real data. Finally, Section 5 is dedicated to discussions and concluding remarks.

2 The MODL co-clustering for two categorical variables

Let X and Y be two categorical variables with values sets $\mathbb{V}^X = \{v_i^X\}$, with $V^X = |\mathbb{V}^X|$ and $\mathbb{V}^Y = \{v_j^Y\}$, with $V^Y = |\mathbb{V}^Y|$. Let $D = \{(x_n, y_n),\ x_n \in \mathbb{V}^X,\ y_n \in \mathbb{V}^Y,\ 1 \leq n \leq N\}$ be a data set with N instances. An example of this data representation is given in Fig. 1, where $\mathbb{V}^X = \{a, b\}$ with $V^X = 2$, $\mathbb{V}^Y = \{A, B, C\}$ with $V^Y = 3$ and $N = 4$.



X	Y
a	B
b	A
b	C
b	A

	A	B	C
a	0	1	0
b	2	0	1

FIG. 1 – *Data representation example.*

The data set D, represented by it's contingency table (see Figure 1), can be summarized using a partition of the values of each variable into clusters/groups. The cross-product of the two partitions of size $I \times J$ forms a $(I \times J)$ co-clustering with one cell per pair of value parts. Note that this method differs from the traditional co-clustering (Govaert and Nadif, 2013) which considers partition of observations and of variables.

In order to choose the "best" co-clustering model $\hat{M}$ (given the data) from the model space $\mathcal{M}$, we use a Bayesian Maximum A Posteriori (MAP) approach. We explore the model space while minimizing a Bayesian criterion, called cost. The cost criterion implements a trade-off between under-fitting and over-fitting and is defined as follows:

$$c(M) = -\log p(M|D) \propto -\log p(M) - \log p(D|M) \tag{1}$$

where $p(M)$ is the prior and $p(D|M)$ is the likelihood of the data given the co-clustering model. The details about the cost criterion and the optimization algorithm (called KHC) are available in Boullé (2011). The key features to keep in mind are: (i) KHC is parameter-free, i.e., there is no need for setting the number of clusters/groups per dimension; (ii) KHC provides an effective locally-optimal solution to the co-clustering model construction, in sub-quadratic time complexity $O(N\sqrt{N}\log N)$ more precisely $O(N^*\sqrt{N^*}\log N^*)$, where $N^* = \sum_{i=1}^{V^X}\sum_{j=1}^{V^Y}\mathbb{1}_{\{n_{ij}>0\}}n_{ij}$ is the number of actual value pairs encountered at least once. However, some data sets come potentially with up to billions of instances and variables having millions of values. These data sets cannot be analyzed using KHC, unless using machines equipped with hundreds of Gb of RAM and still waiting days of computation.

3 Scaled MODL co-clustering

Our objective is to extend the co-clustering optimization algorithms to such large scale data, given the MODL co-clustering criterion, while taking into consideration the following *memory constraints for the Scaled MODL co-clustering*.

— the algorithm can store all the $\mathbb{V}^X$, $\mathbb{V}^Y$ values in the memory,

— the N^* actual pairs of values cannot be stored in memory; and they can only be stored on the disk,

— we can run our co-clustering algorithm on matrices of size at most $I_{max}^2 \ll N^*$.

Finally, the optimized co-clustering model must fit in memory with the following memory complexity $O(V^X) + O(V^Y) + O(I_{max}^2)$.

Suppose, the observed data D can hardly be co-clustered because of one or two following reasons. First, the number of instances N can be very large and second, the number of values on each dimension V^X or V^Y can be too large to be handled by current co-clustering algorithms. To handle this, we consider the memory constraints and propose a two level co-clustering algorithm that allows KHC to produce co-clustering models faster with the smallest possible decrease of their quality. The algorithm is organized in two phases. The first phase consists in the *Split phase* given by the two following steps.

1. *Partitioning step:* aims at obtaining sub data sets from the whole data, such that future co-clustering on each of them meet the memory constraints.
2. *Fine co-clustering step:* builds a co-clustering from each sub data sets using the KHC tool.

The second phase consists in the *Aggregation phase* with the following two steps.

3. *Amalgamate step:* consists in building a global co-clustering on the initial (large) data set by merging the co-clusterings obtained from the sub data sets.
4. *Post-optimization step:* improves the model by the following tracks. First merge clusters and second move values between clusters.

As a consequence, our proposed two level algorithm is a four steps process, that are further described more precisely.

3.1 Split Phase

In our method, we adopt a divide-and-conquer approach, that starts with the split phase.

3.1.1 Partitioning step

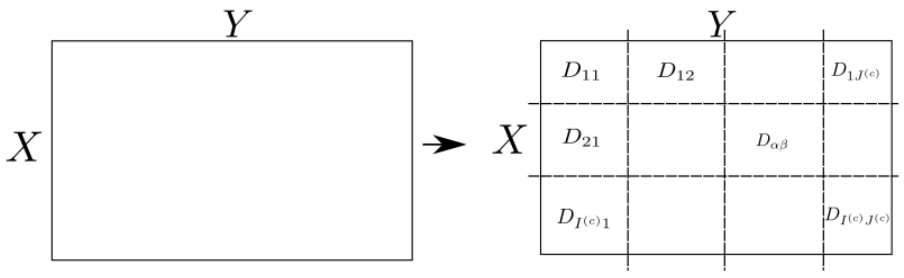


FIG. 2 – *Example of coarse co-clustering of data set D, with* $I^{(c)} \times J^{(c)}$ *coarse co-clusters.*

A large data set, with at least one violated memory constraint, leads us to the very first step of our proposed algorithm, the *Partitioning step*. Within the co-clustering terminology we name it *Coarse co-clustering*. This first step consists in coarse co-clustering D in order to obtain $I^{(c)}$ coarse clusters based on $\mathbb{V}^X$ and $J^{(c)}$ coarse clusters based on $\mathbb{V}^Y$. We obtain $G^{(c)} = I^{(c)} \times J^{(c)}$ coarse co-clusters. Let, $\mathbb{V}^X_\alpha$ and $\mathbb{V}^Y_\beta$ be the sets of values for respectively α and β coarse clusters, such that $\mathbb{V}^X = \bigcup_{\alpha=1}^{I^{(c)}} \mathbb{V}^X_\alpha$ and $\mathbb{V}^Y = \bigcup_{\beta=1}^{J^{(c)}} \mathbb{V}^Y_\beta$.

We propose a random partitioning algorithm, which works as follows. First, we shuffle the values of variables X and Y. Second, we partition the shuffled variable values into respectively $I^{(c)}$ and $J^{(c)}$ parts of equal size. Since our variable values are shuffled, this initial solution is likely to be blind to information patterns. Thus, using such a solution and continue to the next steps can produce a non informative co-clustering result. In order to bypass this issue, a pre-optimization step, similar to Boullé (2011), is used. This pre-optimization step consists in improving the MODL cost (1) by moving the values between clusters, thus improving the initial co-clustering solution by moving the boundaries.

The $G^{(c)} = I^{(c)} \times J^{(c)}$ coarse co-clusters are actually related to sub data sets that are further easier analyzed accordingly to a smaller data size. Note by $D_{\alpha\beta} = \{(x, y) \in D, x \in \mathbb{V}^X_\alpha, y \in$

$\mathbb{V}^Y_\beta\}$ the sub data sets of D, where $1 \le \alpha \le I^{(c)}$ and $1 \le \beta \le J^{(c)}$. Also, $D = \bigcup_{\alpha\beta} D_{\alpha\beta}$. Each of these sub data sets is adapted to the memory constraints. Fig.2 shows an example of coarse co-clustering on the data D.

The complexity of this step is $O(I^{(c)}J^{(c)}(V^X + V^Y)/2)$, thus the computation time grows linearly with $I^{(c)}$ and $J^{(c)}$.

3.1.2 Fine co-clustering step

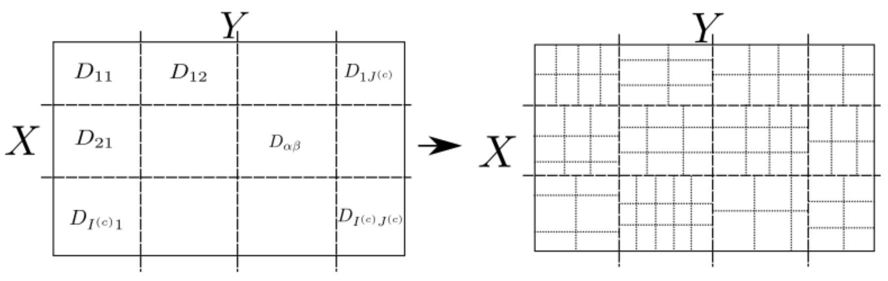


FIG. 3 – *Example of fine co-clustering for each sub data set $D_{\alpha\beta}$ into $I^{(f)}_{\alpha\beta} \times J^{(f)}_{\alpha\beta}$ co-clusters.*

This step consists in running KHC on each of the earlier obtained sub data sets. We name it the fine co-clustering step. On the basis of the fine co-clustering for all sub data sets $D_{\alpha\beta}$, $\forall\alpha = 1, \ldots, I^{(c)}$, $\forall\beta = 1, \ldots, J^{(c)}$, we obtain $I^{(f)}_{\alpha\beta}$ number of fine clusters based on $\mathbb{V}^X_\alpha$ and $J^{(f)}_{\alpha\beta}$ number of fine clusters based on $\mathbb{V}^Y_\beta$. To summarize, we have $G^{(f)}_{\alpha\beta} = I^{(f)}_{\alpha\beta} \times J^{(f)}_{\alpha\beta}$ fine co-clusters for each sub data set $D_{\alpha\beta}$. Fig. 3, shows an example of fine co-clustering for the whole data set D.

Note that this step produces different sized fine co-clusterings, with different fine clusterings, for each sub data set, thus we need to combine all the obtained co-clustering results for the whole data set D.

The complexity of this step is $O\left(N\sqrt{N/I^{(c)}J^{(c)}}\log N/I^{(c)}J^{(c)}\right)$. Observe that, contrarily to the partitioning step, a high number of partitions decreases the computation time of the fine co-clustering step, therefore Section 3.3 is dedicated to show how we choose an optimal number of parts.

3.2 Aggregation phase

In this phase we aggregate the results of the split phase.

3.2.1 Amalgamate step

The amalgamate step, that starts the aggregation phase of our two level algorithm, consists in computing clusters for the entire large data set D by combining all of the obtained fine clusters of the sub data sets. In this step, we refer to the obtained clusters as *micro clusters*.

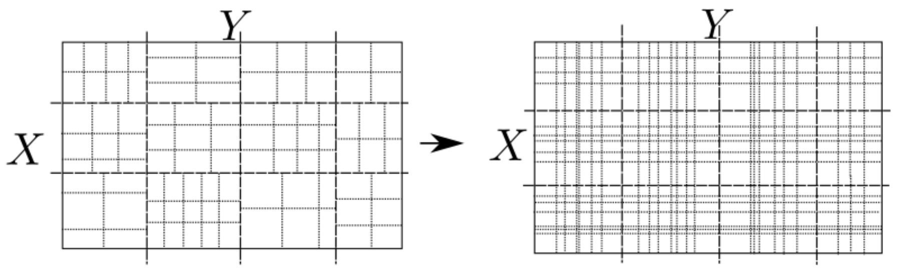


FIG. 4 – *Example of amalgamate of the whole data set D, with $I^{(m)} \times J^{(m)}$ micro co-clusters.*

Succeeding the amalgamate step we obtain $I^{(m)}$ micro clusters based on $\mathbb{V}^X$ and $J^{(m)}$ micro clusters based on $\mathbb{V}^Y$. Fig. 4 illustrates an example of amalgamate step on the entire data set.

3.2.2 Post-optimization step

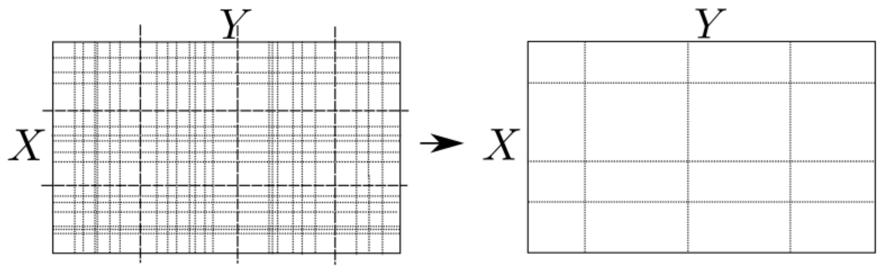


FIG. 5 – *Example of post-optimization of the whole data set D.*

As for the amalgamate step, we need to recall the memory constraint that says that our co-clustering algorithm can run on matrices of size at most I_{max}^2. However, the amalgamate step can eventually produce a too large number of micro clusters with $I^{(m)} > I_{max}$ or $J^{(m)} > I_{max}$. Before proceeding with the post-optimization step, some amalgamate results could be necessary to reduce $I^{(m)}$ and $J^{(m)}$ such that $I^{(m)} \leq I_{max}$ and $J^{(m)} \leq I_{max}$. We propose a sampling approach that consists in randomly grouping the micro clusters into the maximum number of possible clusters I_{max}. This works as follows for each dimension. First, we shuffle the micro clusters and group them into equal I_{max} clusters. Second, to improve the model we move the micro clusters between groups. This results in a randomized co-clustering model that is be further improved by the post-optimization step.

Boullé (2011) proposed two post-optimization types: the *exhaustive merge* and the *greedy post-optimization*. We use similar post-optimization approaches to merge clusters and then move values between clusters.

Merge clusters: consists in merging clusters until the null model is observed. The best co-clustering model is then retained.

Value move: moves the values between the clusters alternatively for each variable.

Fig. 5 illustrates an example of post optimization on the entire data set.

3.3 Choosing the optimal number of parts

One main encountered problem in our proposed two-level co-clustering algorithm is to choose the optimal size of partitions $I^{(c)}$ and $J^{(c)}$ in the partitioning step. We highlight the fact that for small data, KHC tool is more efficient than our two level co-clustering algorithm. This is because the time behavior on the set of processes of our algorithm is greater than the time behavior of one KHC process directly on the whole data set. According to this we assume that per sub data set, we need to have at least 200 values per variable and 10^4 instances.

Let $T = T_S + T_A$ be the global execution time of our two level co-clustering approach, where T_S is the computation time of the Split phase and T_A is the computation time of the Aggregation phase. Our experiments show that T_A is not impacted by the number of partitions, therefore we focus on T_S to minimize T. Recall that T_S is composed of the computational time of the partitioning step, which increases with partition size, and the computational time of the fine co-clustering step, which decreases with partition size. Therefore, to deduce a theoretical proposal for choosing the number of partitions, we use a heuristic approach that equalizes the time complexity of the partitioning step $O(I^{(c)}J^{(c)}(V^X + V^Y)/2)$ with the time complexity of the fine co-clustering step $O\left(N\sqrt{N/I^{(c)}J^{(c)}}\log N/I^{(c)}J^{(c)}\right)$ and assumes that the size of partitions on X and Y ($I^{(c)}$ and $J^{(c)}$) are proportional to their respective number of modalities (V^X and V^Y). We obtain:

$$J_*^{(c)} = \left\lceil c_*\sqrt{V^Y/V^X}\left(\frac{2N\sqrt{N}\log N}{V^X + V^Y}\right)^{\frac{1}{3}}\right\rceil \quad (2)$$

$$I_*^{(c)} = \left\lceil \sqrt{V^X/V^Y}J_*^{(c)}\right\rceil \quad (3)$$

where $c_* = 1/4$ is a constant factor adjusted from our experiments.

4 Experiments

We perform experiments both on simulated and real data in order to evaluate our proposed two-level co-clustering algorithm. In this experiments we run the MODL co-clustering approach on the generated and real world data sets and compare them with our two level co-clustering algorithm, given by 2L-KHC. Indeed, the MODL co-clustering runs in anytime fashion, until no significant changes are observed, and outputs intermediate solutions. Therefore, we show the results of the fist (KHC(1)) and the last (KHC) solutions of the MODL co-clustering. The goal of these experiments is to get a good and simple summary of the data set. We evaluate the quality of the co-clustering model using the normalized cost, computed by $1 - \frac{c(\mathcal{M})}{c(\mathcal{M}_0)}$, where $c(\mathcal{M})$ is the cost of the estimated model and $c(\mathcal{M}_0)$ is the cost of the null model. This normalized cost can be interpreted as a compression rate. Also, in order to show the efficiency of our proposed algorithm we provide the computation time for each approach.

4.1 Experiments on simulated data

In this experiment we first generate our data sets D with two categorical variables X and Y. To generate the data, we use the following probability distribution : $p(x_n = i, y_n = j) \propto 1 - \left|\frac{|(i-j)|}{V}\right|^b$, where (i, j) are the possible categorical values for variable X and Y respectively; $V = V^X = V^Y$ is the number of values for variable X or Y, that for simplicity are considered to be equal; and b is a parameter that controls the concentration of the data simulated on the diagonal of the data matrix.

We vary data mixtures and sparsity by generating three data type families. These are uniform, skewed and sparse families. Fig. 6 shows an example of these three data types families. First, we generate uniform and respectively skewed data families. The values of X are given

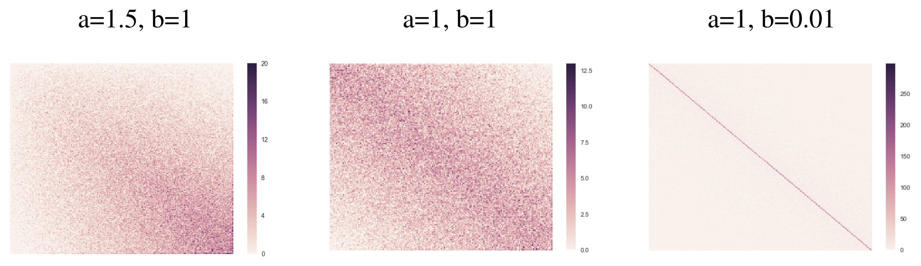


FIG. 6 – *Example plots of skewed (left), uniform (center) and sparse (right) data sets.*

by $i = \lceil u * V \rceil$ and the values of Y are given by $j = \lceil w * V \rceil$, where (u, w) are random variables drawn independently from the power law $ax^{(a-1)}$, with a a shape parameter controlling the balance in our data. For $a = 1$ the uniform data family is generated, while for a higher a skewed data family are generated. For our experiments we fix $a = 1.5$ for to generate skewed data. For a better comprehension of the difference between uniform and skewed data sets, the skewed data generates less data for first generated (i, j) value pairs. Also, setting a small $b = 0.01$ concentrates the data in the diagonal, thus obtaining the sparse data family.

To show the effectiveness of our two level co-clustering algorithm we generate six types of data sets by varying the number of instances and values per variable. Table (1) summarizes the generated data sets.

Dataset	D_1	D_2	D_3	D_4	D_5	D_6
N	10^6	10^6	10^6	10^7	10^7	10^7
V	200	2000	20000	200	2000	20000

TAB. 1 – *Generated data sets.*

First, we run our experiments on uniform data sets. Table 2, shows the obtained normalized cost and the computation time, for KHC and for our two level co-clustering approach. Recall that KHC runs in any time fashion providing intermediate solutions. In our results we present the first and the last retrieved solutions noted by respectively KHC(1) and KHC. Our two level co-clustering approach is given by 2L-KHC. Observe that when the number of values is small, $V = 200$ $(D1, D4)$, the 2L-KHC approach obtains a better solution than that of KHC(1) given the same optimization time. The final solution of KHC is improved by 0.5%, while our approach is ten times faster. For $D2, D3$, $D5$ and $D6$ when the numbers of values per variable are $V = 2000, 20000$ we can see that the 2L-KHC approach obtains a better solution than that of KHC(1) with approximately 15-50 less time. Also, it is approximately 70-150 times faster than KHC, while obtaining models of comparable quality.

	Normalized cost			Time(s)		
Data	2L-KHC	KHC(1)	KHC	2L-KHC	KHC(1)	KHC
D1	0.005354	0.005311	0.005381	8	10	84
D2	0.003270	0.003127	0.003282	277	3885	20324
D3	0	0	0	361	18113	18113
D4	0.005534	0.005525	0.005537	11	11	116
D5	0.003792	0.003718	0.003793	1036	32015	204137
D6	0.002533	0.002447	0.002538	4056	196974	602534

TAB. 2 – *The obtained co-clustering results on uniform data sets.*

Second, we run our experiments on sparse data sets. Table 3, shows the obtained results on our sparse data. Note that, for $V = 200$ ($D1$,$D4$), the 2L-KHC approach obtains the same co-clustering quality as that of the KHC(1) and KHC, while the computation times for all of these approaches are rather small. However, when we have $V = 2000$ values per variable ($D2$, $D5$), we observe that our two level co-clustering approach is $2 - 10$ times faster than KHC(1) and $15 - 60$ faster than KHC, while having a normalized cost 5% worse than that of KHC. Finally, with $V = 20000$ ($D3$, $D6$), our co-clustering approach gives a slightly better solution being 40 times faster than KHC(1) and 80 times faster than KHC.

	Normalized cost			Time(s)		
Data	2L-KHC	KHC(1)	KHC	2L-KHC	KHC(1)	KHC
D1	0.08474	0.08474	0.08474	48	35	342
D2	0.01535	0.01484	0.01587	2282	3939	34326
D3	0.0041	0	0	427	21586	21586
D4	0.08951	0.08951	0.08951	43	23	262
D5	0.01750	0.01749	0.01750	2409	29449	142887
D6	0.01076	0.01045	0.01046	4966	193273	405939

TAB. 3 – *The obtained co-clustering results on sparse data sets.*

Because of lack of space, the skewed data sets results are not shown in this paper. However the obtained results are similar to those of the uniform data.

To conclude, the proposed two level co-clustering approach outperforms KHC in computation time without considerably degrading the quality of the co-clustering solutions.

4.2 Experiments on real world data

We perform experiments on real data which enables us to evaluate our two level co-clustering approach on data with a more complex distribution as compared to the generated data.

4.2.1 Data

We conduct experiments on several real data sets: 20 Newsgroups (Mitchell, 1997), WebSpam (Castillo et al., 2008) and Netflix (Bennett and Lanning, 2007) (with 1% and 10% randomly chosen users), whose characteristics are summarized in Table 4.

Data set	N	V^X	V^Y	Co-clustering variables
20 Newsgroups	2.047.830	19.464	11.315	text $\times$ words
WebSpam	13.068.666	390.130	400.000	source site $\times$ target site
Netflix (1%)	960.327	16.235	4.649	users $\times$ films
Netflix (10%)	10.049.248	17.764	48.068	users $\times$ films

TAB. 4 – *Real world data sets.*

The 20 Newsgroups data set has become popular for experiments in text applications of machine learning techniques, such as text classification and text clustering. It consists of a collection of approximately 20.000 newsgroup documents. This data comprises 2.047.830 observations, 19.464 texts and 11.315 words.

The WebSpam data set comes from a detection challenge of spam type website. The data consists of an extract of the web graph with 13.068.666 links of 390.130 source sites and 400.000 target sites.

The Netflix data set consists of 100 millions of observations, corresponding to the ratings of 480.000 users related to 18.000 films. In order to have faster results, we choose to investigate on approximately 1% and 10% of randomly chosen users. Thus we obtain two data sets. The first one contains 1% randomly chosen users consisting of 960.327 observations with 4.649 users and 16.235 films. The second one contains 10% randomly chosen users and consists of 10.049.248 observations with 48.068 users and 17.764 films.

4.2.2 Results

Table 5 shows the obtained result on the real data sets, where we evaluate the normalized cost and the computation time.

First, observe the results on the 20 Newsgroups. One can see that our two level algorithm is two times faster, with a normalized cost is about 10% worse than that of the first KHC solution, KHC(1).

Next, our results on the WebSpam data set, shows that our two level co-clustering algorithm is two times faster than KHC(1) or 15 times faster than KHC, with a normalized cost within 10% of that of the KHC.

	Normalized cost			Time(s)		
Data	2L-KHC	KHC(1)	KHC	2L-KHC	KHC(1)	KHC
20 Newsgroups	0.0153	0.0163	0.0170	6510	12840	597600
WebSpam	0.2160	0.2331	0.2427	43130	84859	716552
Netflix (1%)	0.0184	0.0190	0.0191	1529	5534	78399
Netflix (10%)	0.0199	0.0202	0.0202	29523	354888	3548888

TAB. 5 – *The obtained co-clustering results on real world data.*

Finally our results on both Netflix data sets show good performance for our 2L-KHC proposed approach. We can see that for the Netflix with 1% of users, the computation of our two level co-clustering algorithm is three times faster than the KHC(1) and 50 times faster than KHC, obtaining 4% worse normalized cost. Also, for the Netflix with 10% of users we see that our two level co-clustering algorithm is about 12 times faster than KHC(1) and about 120 times faster than KHC, losing just 2% of co-clustering quality.

To conclude, experiments on real data show that, our two level co-clustering approach produces faster solutions without considerably decreasing the quality of the co-clustering results. This is especially noticed on data that needs hours of computation with our approach instead of days of computation with KHC. Also, it is noteworthy that our two level co-clustering approach uses much less memory than KHC. For example, the Netflix with 10% of randomly chosen users, requires a machine with at least 10 Gb RAM to run KHC, while our proposed two level co-clustering approach can run on machine with about 1 Gb of RAM.

5 Conclusions and perspectives

In this paper, we have presented a two level co-clustering algorithm using the MODL criterion, that allows processing large data sets that does not fit in memory. The first level, that is the Split phase, consists of the partitioning and the fine co-clustering steps, while the second level, that is the Aggregation phase, consists of the amalgamate and post optimize steps. We have investigated each step of our two level co-clustering algorithm.

To highlight the performance of our two level co-clustering algorithm, we have performed experiments on simulated and real world data. We note that for small data sets, the KHC tool is favorable, however for larger data, the proposed approach is more suitable if we want to obtain a faster solution, without considerably decreasing the quality of the co-clustering solutions.

Finally, in our future work, we will focus on our two level co-clustering algorithm amelioration, for example by parallelizing it. Also, experiments on larger data sets will be investigated.

References

Bennett, J. and S. Lanning (2007). The netflix prize. In *Proceedings of the KDD Cup Workshop 2007*, New York, pp. 3–6. ACM.

Bock, H. (1979). Simultaneous clustering of objects and variables. In *E. Diday (ed) Analyse des données et Informatique*, pp. 187–203. INRIA.

Boullé, M. (2011). Data grid models for preparation and modeling in supervised learning. In I. Guyon, G. Cawley, G. Dror, and A. Saffari (Eds.), *Hands-On Pattern Recognition: Challenges in Machine Learning, volume 1*, pp. 99–130. Microtome Publishing.

Castillo, C., K. Chellapilla, and L. Denoyer (2008). Web spam challenge 2008. In *4th International Workshop on Adversarial Information Retrieval on the Web (AIRWeb)*, Beijing, China.

Dhillon, I. S., S. Mallela, and D. S. Modha (2003). Information-theoretic co-clustering. In *Proc. of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '03, New York, NY, USA, pp. 89–98. ACM.

Govaert, G. and M. Nadif (2008). Block clustering with Bernoulli mixture models: Comparison of different approaches. *Computational Statistics & Data Analysis 52*(6), 3233–3245.

Govaert, G. and M. Nadif (2013). *Co-Clustering* (1st ed.). Wiley-IEEE Press.

Guigourès, R., D. Gay, M. Boullé, F. Clérot, and F. Rossi (2015). Country-scale exploratory analysis of call detail records through the lens of data grid models. In *Proceedings of the ECML/PKDD*, pp. 37–52. Springer International Publishing.

Hartigan, J. A. (1972). Direct Clustering of a Data Matrix. *Journal of the American Statistical Association 67*(337), 123–129.

Li, H. and N. Abe (1998). Word clustering and disambiguation based on co-occurrence data. In *Proc. of the 17th International Conference on Computational Linguistics - Volume 2*, COLING '98, Stroudsburg, PA, USA, pp. 749–755. Ass. for Comp. Linguistics.

Mechelen, I. V., H. H. Bock, and P. D. Boeck (2004). Two-mode clustering methods: a structured overview. *Statistical methods in medical research 13*(5), 363–394.

Mitchell, T. M. (1997). *Machine Learning* (1 ed.). New York, NY, USA: McGraw-Hill, Inc.

Papadimitriou, S. and J. Sun (2008). Disco: Distributed co-clustering with map-reduce: A case study towards petabyte-scale end-to-end mining. In *ICDM*, pp. 512–521. IEEE Computer Society.

Résumé

La classification croisée (co-clustering) est une technique qui permet d'extraire la structure sous-jacente existante entre les lignes et les colonnes d'une table de données sous forme de blocs. Plusieurs applications utilisent cette technique, cependant de nombreux algorithmes de co-clustering actuels ne passent pas à l'échelle. Une des approches utilisées avec succès est la méthode MODL, qui optimise un critère de vraisemblance régularisée. Cependent, pour des tailles plus importante, cette méthode atteint sa limite. Dans cet article, nous présentons un nouvel algorithme de co-clustering à deux niveaux, qui compte tenu du critère MODL permet de traiter efficacement de données de très grande taille, ne pouvant pas tenir en mémoire. Nos expériences montrent que l'approche proposée gagne en temps de calcul tout en produisant des solutions de qualité.