Analysis and Experimental Evaluation of a simple Algorithm for Collaborative Filtering in Planted Partition Models (Extended Abstract)

Devdatt Dubhashi[1] and Luigi Laura[2] and Alessandro Panconesi[3]

[1] Chalmers University of Technology – Dept. of Computing Science
SE-41296 Gothenburg, Sweden
dubhashi@cs.chalmers.se
[2] Universitá di Roma "La Sapienza" – Dip. di Informatica e Sistemistica
Via Salaria 113, Roma
laura@dis.uniroma1.it
[3] Universitá di Roma "La Sapienza" – Dip. di Informatica
Via Salaria 113, Roma
ale@dsi.uniroma1.it

Abstract. We introduce a random planted model of bi-categorical data to model the problem of collaborative filtering or categorical clustering. We adapt the ideas of an algorithm due to Condon and Karp [4] to develop a simple linear time algorithm to discover the underlying hidden structure of a graph generated according to the planted model with high probability. We also give applications to the probabilistic analysis of Latent Semantic Indexing (LSI) in the probabilistic corpus models introduced by Papadimitriou et al [12]. We carry out an experimental analysis that shows that the algorithm might work quite well in practice.

1 Introduction

Recommendation systems that are based on simultaneous clusterings across several categories and an overlap of interests is called *collaborative filtering* since selection of items is done in a manner resembling individuals collaborating to make recommendations to their friends. Collaborative filtering methods have been applied in many areas both in research [14, 11, 2, 10] and in industry [7, 15]. In spite of a wealth of tools the problem remains challenging and much work remains to be done before a satisfactory solution can be found. In particular, besides the ever present need for fast and simple algorithms that work well in practical scenarios, perhaps as importantly, there is also a need for appropriate theoretical tools that can assist in the development of good algorithms.

In this paper, we develop a natural model of categorical data based on the so-called *planted partition models* [8, 4]. In our theoretical analysis, we restrict ourselves, as a first step, to two categories , called "people" and "movies". Roughly, we have a set of "people" P partitioned into two unknown sets P_1 and P_2 and likewise a set M of "movies" partitioned into unknown sets M_1 and M_2. While there are many (random) edges between M_i and P_j when $i = j$, there are very few (random) connections between M_i and P_j when $i \neq j$. We seek an algorithm that, if fed with a bipartite graph of this type, is able to identify the unknown clusters P_1, P_2 and M_1, M_2.

In this paper we describe a very simple randomized algorithm to discover the hidden bi-categorical clustering structure in the quasi-random graphs generated according to this model (formally defined in § 2). We analyze the algorithm to show that it succeeds in discovering the underlying structure *exactly* with high probability. The running time of the algorithm is linear.

While in the theoretical analysis we restrict ourselves to the 2 by 2 case, in our experiments we also test the algorithm in the k by k case, for $k = 2, 3, 4, 5$ (k clusters on each side). The results are very promising and show that the algorithm might work well in practice. In particular, for $k > 2$, rather than applying the natural recursive approach, we exploit a concentration of measure phenomenon first observed in [4] and that seems to apply in our new context too. Namely, given the input graph, we compute a 2 by 2 partition. Let P_1, P_2 and M_1, M_2 be the clusters output by the algorithm. To isolate the k clusters of type "people" for instance, we process the vertices one by one and group them in a greedy fashion according to the quantity

$$\#\text{edges from the vertex to} M_1 - \#\text{edges from the vertex to} M_2$$

This random variable is sharply concentrated, so that we obtain k clumps from which the k true clusters can be easily obtained. The same process can be appied on the movie side. We remark that while concentration of measure is proven in [4], this is not done here for our bipartite scenario. The experiments seem to indicate however that the phenomenon could very well hold.

Our quasi-random models are admittedly not entirely realistic, but we believe they form a very relevant test-bed for solutions because (a) they capture a very important aspect of the problem, namely the bicategorical clustering and (b) they admit much flexibility and variations towards more realistic models.

The work in this paper answers the open question posed in [4] whether their ideas can be used to address the collaborative filtering or categorical clustering problem. While very similar to their algorithm, ours incorporates a few modifications relevant to the bicategorical case. The analysis of the algorithm also follows closely the analysis of [4], but with one significant new ingredient. While the analysis of [4] can be viewed as a discrete probabilistic analogue of a differential equation that governs the underlying stochastic process, our analysis is a discrete probabilistic analogue of a *coupled* system of two differential equations.

In the last section of the paper, we also show that similar ideas are also relevant in information retrieval in the context of probabilistic models and analysis of Latent Semantic Indexing (LSI), the foundations of which are laid out in [12]. Our analysis indicates that in these cases where traditional spectral methods for LSI [12, 1] works well, one might achieve even better results more efficiently by applying very simple probabilistic algorithms. We show that our collaborative filtering or categorical clustering algorithm can outperform traditional LSI in these probabilistic corpus models.

1.1 Relation to Past Research

Our work falls in the area of graph partitioning and clustering algorithms, both very well-studied topics with a large literature. Our work is most closely related to the papers of Jerrum and Sorkin [8] and Condon and Karp [4]. The former explored how well the well-known simulated annealing heuristic (applied at constant temperature as the Metropolis algorithm) performs in the graph partitioning problem. They introduced the planted partition model of a random graph (discussed in in § 2) and showed that the Metropolis algorithm discovered this structure with high probability in time $O(n^2)$. Condon and Karp developed simpler algorithms for the problem using a stochastic process inspired by the Metropolis algorithm of Jerrum and Sorkin. They extended the planted partition model to one with several groups and showed that a simple randomized algorithm could discover the planted structure with high probability in linear time.

There are two other papers that are closely related to our work. Ungar and Foster [16] also develop models of random graphs that are similar to our planted partition models. They adopt a statistical approach and use a Gibbs sampling algorithm similar to the well-known EM algorithm to estimate the underlying hidden parameters. Our algorithm is much simpler than theirs and also provably more efficient. While [16] can only invoke a general convergence result in the limit, and give no guarantees on the running time, we prove that our algorithm succeeds in finding the exact partition in linear time with high probability.

Gibson, Kleinberg and Raghavan [6] address the general categorical clustering problem using a completely different approach. They describe an iterative weighting algorithm that can be viewed as a higher order dynamical system. Because much is unknown about the properties of higher-order dynamical systems – in particular their convergence rates and even the limit – they are able to prove only some weak and limited results. They give some convergence results in these special cases, but even here, they are not able to give any guarantees on run times. They report some experimental studies on quasi-random data similar to ours, but give no theoretical analysis of the results.

One of the recent approaches to the problem is [6] and our work offers an alternative approach based on completely different methods. It indicates that simple and efficient stochastic algorithms can be very effective in the context of categorical clustering in practice. Theoretically, our approach is demonstrably superior because in contrast to [6], we are able to prove rigorous and quite general results about the quality and the efficiency of the algorithms.

2 Planted Models

The so-called *planted partition models* [8, 4] are models of random graphs that are the natural setting for the study of partitioning algorithms. In the *planted bisection* model, one has a bipartition with $n/2$ nodes on each side (with even n), which we can refer to as blue and green vertices. An edge is placed between like-colored vertices with probability p and between differently-colored vertices with probability $r < p$. The reason for focussing on this model is that if $p-r > n^{-1/2+\epsilon}$ for any fixed $\epsilon > 0$, then, with probability $1 - \exp(-n^{\Omega(\epsilon)})$, the planted bisection is the unique bisection with minimum cut size. Hence we can demand of a bisection algorithm that given a random graph generated in this model (without revealing the planted bisection, of course), the algorithm must zero in on this unique minimum cut.

A natural extension [4] is the *planted ℓ-partition* model, where we have ℓ groups of n/ℓ vertices each (with ℓ dividing n), each group corresponding to one of ℓ distinct colors. As before, an edge is placed between like-colored vertices with probability p and between differently-colored vertices with probability $r < p$. One can show that if $p - r > n^{-1/2+\epsilon}$ for any fixed $\epsilon > 0$, then, with probability $1 - \exp(-n^{\Omega(\epsilon)})$, the planted partition by color classes as above is the unique multipartition (into ℓ classes) with minimum multi-cut size. Hence we can demand of a partitioning algorithm that given a random graph generated in this model (without revealing the planted partition, of course), the algorithm must zero in on this unique minimum multicut.

There is a natural extension of these models to our scenario with two categories of data, which we will think of as people (P) and movies (M). We will refer to these as *bicategorical planted models*. In the bisection model, we have n people and m movies, both divided into two equi-sized classes corresponding to two colors (so both n, m are even). In the ℓ-partition bicategorical model, both people and movies are divided into ℓ equi-sized classes corresponding to the colors $[\ell]$. In both cases, an edge is placed between a person and a movie of the same color with probability p and between a person and movie of different colors with probability $r < p$. Denote the resulting random bipartite graph by $(P.M, E)$.

We will assume that $n \geq m$. In both the (bicategorical) bisection and mutipartition case, it can be shown that if $p - r > m^{-1/2+\epsilon}$ for any fixed $\epsilon > 0$, then, with probability $1 - \exp(-m^{\Omega(\epsilon)})$, the planted (bicategorical) partition is the unique one which minimizes the number of non-monochromatic edges. We will assume henceforth that $\Delta := p - r > m^{-1/2+\epsilon}$ for some fixed $\epsilon > 0$.

3 The Algorithm

In this section, we present our linear-time algorithm for the collaborative filtering problem with two categories ("people" and "movies") and two classes or groups in each category. (In § 5 we indicate how the algorithm can be extended to several classes in each category.)

The algorithm runs in four phases. The purpose of the first two phases is to build a *statistically accurate* partition of the *smaller* category ("movies") i.e. a partition in which the two parts of the partition are almost monochromatic: almost all vertices are of the same color with a few vertices of the opposite color. The third phase will use this statistically accurate partition to construct an *exact* partition of the other category ("people"). Finally, the fourth phase will use the exact partition of the first category ("people") to construct an *exact* partition of the second category ("movies").

Phase 1 In this phase, the algorithm will incrementally and *simultaneously* build a partition of both categories: (L_1^P, R_1^P) will be a partition of "people" and (L_1^M, R_1^M) will be a partition of "movies". Initially, both partitions are empty, and they are built up by interleaving the addition of two "people" vertices and two "movie" vertices into the two partitions. In each of $t_1 := m^{1-\epsilon/2}$ steps, choose a pair of "people" nodes (p_1, p_2) and a pair of "movie" nodes $(m_1.m_2)$, picked randomly and uniformly from the unexamined nodes. To classify the "people" nodes, we examine the number of edges from p_1 and p_2 to the partition of the "movies". Let ℓ_i^P, r_i^P be the number of edges from vertex p_i to L_1^M and R_1^M respectively, for $i = 1, 2$. Let $X := \ell_1^P - r_1^P - \ell_2^P + r_2^P$. If $X > 0$, place p_1 in L_1^P and p_2 in R_2^P and if $X < 0$, place them in the opposite manner. If $X = 0$, place them equiprobably into L_1^P and R_1^P. Similarly use the "peoples" partition to place the movie nodes in the appropriate parts of the "movies" partition.

Phase 2 In this step, we build a partition (L_2^M, R_2^M) of "movies". Initially this is empty. Choose $m/4$ new pairs of "movie" vertices chosen uniformly at random from the unexamined vertices and assign them greedily to L_2^M or R_2^M using the "peoples" partition (L_1^P, R_1^P) as in the first phase. Note that all pairs may be processed concurrently.

Phase 3 In this phase, we build the final partition of the "people", (L^P, R^P). Once again, this is built up starting from the empty partition. We repeatedly pick a "people" node from all the "people" vertices (including the ones processed in the first phase) and assign it greedily using the "movies" partition (L_2^M, R_2^M): it is added to L^P if it has more edges to L_2^M than to R_2^M.

Phase 4 Finally we build the final partition of the "movies", (L^M, R^M). This is built up starting from the empty partition. We repeatedly pick a "movie" node from all the "movie" nodes (including those processed in the earlier phases) and assign it greedily using the "peoples" partition (L^P, R^P) symmetrically as in Phase 3.

4 Analysis

4.1 Overview

The progress of the algorithm is measured as in [8, **?**] by tracking the *color imbalance* of the partitions. The color imbalance of a partition is one half of the number of vertices of one color in one half of the partition minus the number of vertices of the same color in the other half. In our case, there are two color imbalances to track: k^P, the color imbalance of the "peoples" partition and k^M, the color imbalance of the "movies" partition. The analysis proceeds by showing that each of the following statements are true with high probability, by which we mean throughout, probability $1 - \exp(-n^{\Omega(\epsilon)})$.

- At the end of Phase 1, both color imbalances k^P, k^M are at least $m^{1-\epsilon}$.
- At the end of Phase 2, the color imbalance k^M is $\Theta(m)$.
- At the end of Phase 3, no node in L^P has the same color as a node in R^P i.e. (L^P, R^P) is a perfect partition of the "people".
- At the end of Phase 4, no node in L^M has the same color as a node in R^M i.e. (L^M, R^M) is a perfect partition of the "movies".

4.2 A Coupling Argument

The analysis of the evolution of the color imbalances k^P, k^M is complicated by the fact the steps in the underlying stochastic processes are not independent and the transition probabilities depend on the history of the process. To overcome this difficulty, we employ the coupling argument from [4][Lemma 4] or [8][Theorem 6.2] which relates the behavior of the process to a simple random walk with independent increments. Thus we can focus on analyzing the evolution of k^P and k^M as a simple random walk on the line.

4.3 Phase 1

At the end of Phase 1, we claim that both the imbalances k^P and k^M are at least $m^{1-\epsilon}$ with high probability. The proof analyzes how the imbalances grow with time. The following claim is the key to the analysis and is very similar to Claim 2 in [4]. It shows that

(b) At every iteration of the greedy partition building process, both imbalances are at least as likely to increase as to decrease.
(c) *The two imbalances mutually reinforce each other*: the higher k^M is, the more likely is k^P to increase and vice-versa. This is a key additional insight over the Condon-Karp analysis.

Proposition 1. *Let $k^P = k^P(t)$ and $k^M = k^M(t)$ denote the imbalances in the partitions (L_1^P, R_1^P) and $L_1^M, R_1^M)$ respectively at time step t of Phase 1. Let k denote either k^P or k^M. Then, at the next time step, for any history of the process till time t,*

(a) $\Pr[k \quad \textit{increases}] = \Omega(1)$.
(b) $\Pr[k \quad \textit{increases}] - \Pr[k \quad \textit{decreases}] \geq 0$.
(c) If $k^P, k^M = \Omega(m^{1/2-\epsilon/2})$, *then*

$$\Pr[k^P \quad \textit{increases}] - \Pr[k^P \quad \textit{decreases}] = \Omega(\min(k^M \Delta/\sqrt{t}, 1)),$$

and

$$\Pr[k^M \quad \textit{increases}] - \Pr[k^M \quad \textit{decreases}] = \Omega(\min(k^P \Delta/\sqrt{t}, 1)),$$

The proof of this proposition is exactly the same as the proof of Claim 2 in [4] using the quantitative version of the Central Limit Theorem called Esseen's inequality [13][Theorem 3, p.111].

The analysis of Phase 1 is now completed by dividing the evolution into two parts:

- In the first part, the evolution of both k^P and k^M can be treated as an unbiased random walk by Proposition 1, part (b). Thus from standard results on the unbiased random walk [5][XIV.3], the expected time for the random walks to reach $m^{1/2-\epsilon/2}$ starting at 0 is $m^{1-\epsilon}$. Applying Markov's inequality, the walk reaches $m^{1/2-\epsilon/2}$ in $O(m^{1-\epsilon})$ steps, and hence the probability that it reaches $m^{1/2-\epsilon/2}$ in $O(m^{1-\epsilon/2})$ steps is $1 - \exp(-\Omega(m^{\epsilon/2}))$.
- In the second part, both random walks now have a distinct *drift* towards increasing: the difference between the probability of an increase and a decrease for the random walk of k^P when the random walk for k^M is at position i is by Proposition 1, part (c), $\delta = \Omega(\min(i\Delta/\sqrt{t}, 1)) = \Omega(\min(i\Delta/\sqrt{t_1}, 1))$. Similarly for k^M. By a standard result on biased random walks [5][XIV.3], the expected time for each of k^P, k^M to double is $O(i/\delta) = O(m^{1-\epsilon})$ and so with high probability, it doubles in time $O(m^{1-3\epsilon/4})$ steps. Thus after $t_1 = m^{1-\epsilon/2}$ steps, both random walks reach $2m^{1-\epsilon}$ with high probability. A similar analysis shows that once this imbalance is reached, it remains at least half this value with high probability.

4.4 Phase 2

In this phase, only the "movies" partition evolves. Each pair of nodes (m_1, m_2) contributes $1/2, 0$ or $-1/2$ to k^M. This contribution is determined by the value of k^P, which, from the analysis of Phase 1 is at least $m^{1-\epsilon}$. Now we use Claim 7 from [4]:

Proposition 2. *Suppose the imbalance k^P at the end of Phase 1 is at least $m^{1-\epsilon}$. Let (m_1, m_2) be a pair of "movie" vertices examined in Phase 2. Then*

$$\Pr[(m_1, m_2) \quad \textit{increases} \quad k^M] - \Pr[(m_1, m_2) \quad \textit{decreases} \quad k^M] = \Omega(1).$$

Thus the value of k^M at the end of Phase 2 dominates $\sum_{1 \leq i \leq m/4} Z_i$ where the Z_i are independent random variables taking values in the set $\{1/2, 0, -1/2\}$ with $\Pr[Z_i = 1/2] - \Pr[Z_i = -1/2] = \Omega(1)$ for each i. Thus $k^M = \Omega(m)$ with probability at least $1 - \exp(-\Omega(m))$ at the end of Phase 2.

4.5 Phase 3

This phase gives rise to an exact partition of the "people" using the statistically-accurate partition of the "movies" produced in Phase 2. For a people's node p, let $\ell_2(p)$ denote the number of edges from p to L_2^M. As in Claim 9 of [4], it follows by a simple application of the concentration of measure that with high probability,

$$|\ell_2(p) - E[\ell_2(p)]| \leq m^{-1/2+\epsilon/2}.$$

Next, for vertices p and p' of opposite colors, a simple calculation shows that

$$|E[\ell_2(p)] - E[\ell_2(p')]| = k^M \Delta = \Omega(m^{-1/2+\epsilon}).$$

Thus the values of $\ell_2(p)$ are sharply clustered around these two widely separated values, and so with high probability, all "people" nodes are correctly partitioned in Phase 3.

4.6 Phase 4

The analysis of this phase is exactly the same as the analysis of Phase 3, with the roles of "people" and "movies" reversed (the additional fact that the imbalance k^P now equals the maximum possible possible value, $n/2$ only helps). Thus with high probability, the "movie" nodes are correctly partitioned in this phase.

Theorem 1. *As long as $p - r \geq m^{-1/2+\epsilon}$ for some constant $\epsilon > 0$, the algorithm applied to a graph generated by the bi-categorical planted model with two classes in each category, discovers the underlying hidden planted structure exactly with probability $1 - \exp(-m^{\Omega(\epsilon)})$.*

5 Extensions

5.1 Multi-planted models

The algorithm and analysis can be extended to the more general case of the multi-planted models where each category is divided into $\ell \geq 2$ color classes as in [4]. In the recursive algorithm, we first split both categories into two parts as above, and then recurse on the two sub-parts. In the non-recursive version of the algorithm, we modify Phase 3 above. The statistically accurate partition (L_2^M, R_2^M) built in Phase 2 has the the additional property that the various color imbalances are well-separated from each other. This can be used to produce an exact partition of the "people" nodes by examining the number of edges to one half of the partition as in [4]. In our experiments below we exploit this fact.

5.2 Noisy data models

One can define a model of quasi-random planted data that also incorporates noise: in addition to the edges defined in the planted model, we also introduce random edges between "people" and "movies" placing an edge between a pair of people-movie nodes independently with probability δ. As long as the noise parameter δ is small, specifically, $\delta = o(p - r)$, the algorithm is unaffected and continues to work correctly. In this sense our algorithm is robust to noisy data.

5.3 Unequal class sizes

It is fairly clear from the description and analysis above that the algorithm and its analysis are fairly robust. In particular they do not really depend on each color class in the planted model being of the same size. The algorithm and analysis will continue to hold as long as the ratio of the largest group to the smallest group is not too large. While we defer the full analysis to the final version, here we give some good empirical evidence indicating that the algorithm performs well in this case too.

6 Experimental results

The input graphs for our experiments are random bipartite graphs of the form $G = (P, M, E)$ that we shall now describe.

- P is the set of "people", $P = P_1 \cup P_2$, where P_1 is the subset of P of color 1 and P_2 is the subset of P of color 2
- M is the set of movies, $M = M_1 \cup M_2$, where M_1 is the subset of M of color 1 and M_2 is the subset of M of color 2
- $|P| = n$, $|M| = m$, and $|P_1| = |P_2| = \frac{n}{2}$, $|M_1| = |M_2| = \frac{m}{2}$.
- p is the probability of like-colored edges, that is the probability of an edge between elements of P_1 and M_1 and elements of P_2 and M_2.
- r is the probability of different-colored edges, that is the probability of an edge between elements of P_1 and M_2 and elements of P_2 and M_1.
- The output of the algorithm are four sets L^P, R^P, L^M, R^M such that $P = L^P \cup R^P$ and $M = L^M \cup R^M$.

Let us assume, without loss of generality, that color 1 is the dominating color in the set L^P (if it is not, we can swap L^P with R^P and L^M with R^M). To measure the quality of the result we shall make use of the following quantities:

- L_1^P is the percentile presence of color 1 in the set L^P, or, more formally,

$$L_1^P = \frac{|L^P \cap P_1|}{|L^P|} \cdot 100.$$

  Similarly,

$$R_2^P = \frac{|R^P \cap P_2|}{|L^P|} \cdot 100,$$

  and

$$L_1^M = \frac{|L^M \cap M_1|}{|L^M|} \cdot 100,$$

  and

$$R_2^M = \frac{|R^M \cap M_2|}{|R^M|} \cdot 100.$$

- With μ we denote the average of the previous quantities, i.e., $\mu = \frac{L_1^P + R_2^P + L_1^M + R_2^M}{4}$

Thus, the 4 parameters $\{m, n, p, r\}$ fix a specific distribution of input bipartite graphs.

In our experiments we are interested in speed and accuracy, for the 2 by 2 as well as the k by k case.

Our experimental results for the accuracy in the 2 by 2 are summarized in Fig.1 through 3. Every "point" in the plots is the average taken over 100 different runs of the algorithm: we generated 10 bipartite graphs and for each of them we ran the algorithm 10 times. Unless stated otherwise, these are the values for the parameters: $n = 100, m = 80, p = 0.7, r = 0.1$.

Fig.1 shows the performance of the algorithm against $p - r$, the probability of like-colored edges minus the probability of differently-colored edges. Specifically r is kept fixed at 0.1, while p increases from 0.3 to 0.9. On the y-axis we report the "quality" of the partitioning made by the algorithm. For each value of $p - r$ there are 4 plots, corresponding to L_1^P, R_2^P, L_1^M and R_2^M. Notice that these value quickly rise from 60% (mediocre accuracy) to over 90% (excellent accuracy), as $p - r$ increases.

In Fig.2 we keep the size of the movie set fixed and increase the size of the people set. In this way we get an indication of performance of the algorithm when the relative sizes differ. Here $|P| = n = \alpha m = \alpha|M|$,

$m = 100$ is kept fixed while $\alpha = 1, 2, \ldots, 10$. Here, $p - r = 0.6$. On the x-axis we have α while, as before, on the y-axis we have L_1^P, R_2^P, L_1^M and R_2^M.

As it can be seen the algorithm is fairly robust and its performance does not depend on α and hence on the relative sizes of the sides of the bipartition.

Finally in Fig.3 we analyze "scalability". That is, we blow the sizes of both sides of the bipartition. Here, $|P| = n = \beta n^*$ and $|M| = m = \beta m^*$, where $n^* = 100$,$m^* = 80$ and β goes from 1 to 10. As with the previous test, and consistently with the analysis, the quality of the result doesn't depend on the scaling factor β: all the percentiles are over 90% that means that the size of the sets does not influence the quality of the results (provided that they are big enough). We plot L_1^P, R_2^P, L_1^M and R_2^M versus β.

Let us now switch to the k by k case. Recall that the algorithm proceeds as follows. We first compute a 2 by 2 partition M_1, M_2 and P_1, P_2. The algorithm knows k, the number of clusters on each side. To partition the set of people into k clusters we compute for each vertex the quantity

$$\#\text{edges from the vertex to} M_1 - \#\text{edges from the vertex to} M_2.$$

For a given true cluster P_i, these quantities happen to be highly concentrated and, for different P_i and P_j, quite spread apart. It is therefore easy to see k "clumps" corresponding to the k clusters. The same approachy is used to cluster the movies.

Our results are presented in Table 1. The parameters have the following values: $n = m = 100,000$ nodes on each side of the bipartition, and $r = 0.1$. The table shows the results for different values of p, the probability of like coloured edges.

The algorithms displays the following curious behaviour. To fix the ideas, consider the case $k = 4$. Table 1 for $p = 0.7$ reports the following figures:

$$(1,1,1,1) = 4\%$$
$$(2,1,1,0) = 12\%$$
$$(2,2,0,0) = 84\%.$$

This means that only 4 per cent of the times the algorithm was able to produce the 4 correct clusters. 12 per cent of the times the algorithm was able to produce 3 clusters, one of which containing 2 true clusters, and the remaining two containing one true cluster each. Finally, 84 per cent of the times, the algorithm produces just 2 clusters, each containing 2 true clusters together. Strange as it may seem first this does not imply that the algorithm is working poorly!

First, the algorithm might clump true clusters together in one bigger cluster, but it *never* splits a true cluster into two or more. Second, the algorithm works very well in the 2 by 2 and 3 by 3 case (see table), so if bigger clusters are produced they can be separated by running the algorithm again.

Finally, we come to the running time. The results are shown in Figure 4, reporting an implementation with external memory due to the very large sizes we tested the algorithm with. As it can be seen the algorithm is quite fast.

In what follows we describe some of the implementation issues. Our impementation is an example so-called *semi-external graph algorithms*, i.e., algorithms that store information about the nodes in main memory and keep the edges in secondary memory. We decided to have a redundant representation of the graph in secondary memory, and we store twice the edges list, one sorted primarily by people and secondarily by movies, and the other in the opposite way. Every edge list is represented in secondary memory with three arrays: one (the main) contains the list of the pointed nodes, and the other two (the indexes) give, for every pointing node, the outdegree and the starting position of its edges in the first array. So, for the first edge list (sorted by people) we store $2n + \#(edges)$ items, and, similarly, for the second list we use $2m + \#(edges)$ items. The total disk space is therefore $2n + 2m + 2 \cdot \#(edges)$ items, where every item stores a single node. This representation occupies only $2n + 2m$ more items if compared to the simple edge list (where we store 2 items for each edge), but in this way we can compare two sets in a linear time, and we use this property in all the phases of the algorithm (except the first one). We know that, in the *Data Stream* model, such a representation that divides the data could sound surprising, but the index arrays (the ones that store the outdegree and position of pointed nodes) are buffered and the streamed data is the main array.

We also note that a graph generated according to the model we described has an expected number of edges equal to $E = n \cdot (p \cdot \frac{m}{k} + r \cdot (k-1) \cdot \frac{m}{k})$. So, if $m = n = 100,000$, $k = 4$, $p = 0.7$ and $r = 0.1$ we have more than 2 billion edges. We use 12 bytes per node in the index arrays (4 for the outdegree and 8 for the starting position in the main array) plus 4 bytes for each node in the main array. So, with the above quantities, we use approximately 40 GB of hard disk space.

7 Application: Probabilistic Analysis of LSI

Papadimitriou *et al* [12] took a first step in developing a probabilistic model and analysis to explain the effectiveness of LSI in diverse applications. They introduced a probabilistic corpus model of generating data. There is an underlying *universe* of all terms, U. A *topic* is a probability distribution on U. A meaningful topic is very different from the uniform distribution on U; usually it will be concentrated on a small set of terms that are highly relevant to a particular subject. A corpus is said to be *ϵ-separable* (for $0 \leq \epsilon < 1$) if each topic T in the corpus is associated with a subset of terms U_T, called the *primary terms* of T such that $U_T \cap U_{T'} = \emptyset$ for different topics T, T' and each topic T concentrates probability at least $1 - \epsilon$ on U_T. A *pure corpus model* is a probability distribution $\mathcal{D}$ on pairs (T, ℓ) where T is a topic and ℓ is a positive integer. The corpus model will be called *uniform* if

- For any two topics T, T', the set of primary terms has the same size: $|U_T| = |U_{T'}|$.
- The distribution $\mathcal{D}$ is uniformly distributed on all topics.

A document is generated form such a corpus by a two-stage process: first a pair (T, ℓ) is picked according to $\mathcal{D}$. Then terms are sampled ℓ times according to the distribution T. A corpus of size n is formed by repeating this two-stage sampling process m times to form n documents.

Let us look at the random bipartite graph between documents and terms generated by such a process of data generation. It is clear that the resulting graph is very similar to a planted partition model, with partitions corresponding to topics on the topic side and to documents belonging to a particular topic on the document side. The only difference is that not all edges form a document to a term within its primary set of terms have the same probability. However, a similar analysis shows that the same algorithm succeeds with high probability in recovering the correct partitions.

Theorem 2. *The collaborative filtering algorithm will be* 0*-skewed on a pure uniform* ϵ*-separable corpus model as long as* $1 - 2\epsilon \geq n^{-1/2+\epsilon}$.

It seems likely that our methods will work well for the corpus model in [12] and also more realistic generalizations of it. We hope to develop this in future work.

Acknowledgements

We wish to thank Olle Häggström and Jeff Steif for valuable discussions.

References

1. Yossi Azar, Amos Fiat, Anna R. Karlin, Frank McSherry, and Jared Saia, “Spectral Analysis of Data”, STOC 2001.

	$k=3$	$k=4$	$k=5$
$p=0.7$	$(1,1,1)=83\%$	$(1,1,1,1)=4\%$	$(1,1,1,1,1)=0\%$
	$(2,1,0)=17\%$	$(2,1,1,0)=12\%$	$(2,1,1,1,0)=0\%$
		$(2,2,0,0)=84\%$	$(2,2,1,0,0)=70\%$
			$(3,2,0,0,0)=30\%$
$p=0.8$	$(1,1,1)=74\%$	$(1,1,1,1)=3\%$	$(1,1,1,1,1)=0\%$
	$(2,1,0)=26\%$	$(2,1,1,0)=17\%$	$(2,1,1,1,0)=0\%$
		$(2,2,0,0)=80\%$	$(2,2,1,0,0)=63\%$
			$(3,2,0,0,0)=37\%$
$p=0.9$	$(1,1,1)=80\%$	$(1,1,1,1)=0\%$	$(1,1,1,1,1)=0\%$
	$(2,1,0)=20\%$	$(2,1,1,0)=10\%$	$(2,1,1,1,0)=0\%$
		$(2,2,0,0)=90\%$	$(2,2,1,0,0)=84\%$
			$(3,2,0,0,0)=16\%$

Table 1. Results for $k > 2$ clusters.

2. M. Balabanovic and Y. Shoham, "Fab: Content-Based Collaborative Recommendation", *Comm. ACM*, 40:3, pp. 66–72, 1997.
3. Barbour, A.D., L. Holst and S. Janson, *Poisson Approximation*, Oxford University Press 1992.
4. Condon, A. and R. Karp, "Algorithms for Graph Partitioning on the Planted Partition Model", *Random Str. and Algorithms*, 18, pp. 116–140, 2001.
5. W. Feller, *An Introduction to Probability Theory and its Applications*, 3rf ed. Vol. 1, Wiley 1968.
6. D. Gibson, J. Kleinberg and P. Raghavan, "Clustering Categorical Data: An Approach Based on Dynamical Systems", *Proc. of the 24th VLDB Conference*, 1998.
7. Goldberg D., D. Nichols, B. Oki and D. Terry, "Using collaborative filtering to weave an information tapestry", *Comm. ACM*, 35:12, 61–70, 1992.
8. Jerrum, M. and G.B. Sorkin, "The Metropolis Algorithm for Graph Bisection", *Discrete Appl. Math.*, 82 (1–3), pp. 155-175, 1998.
9. Henzinger, M.R., Raghavan, P. and S. Rajagopalan. "Computing on data streams", SRC Technical Note 1998-011. Also in DIMACS series in Discrete Mathematics and Theoretical Computer Science, volume 50, pages 107–118, 1999.
10. Konstan, J.A., B.N. Miller, B. Maltz et al, "Grouplens: Applying collaborative filtering to Usenet news", *Comm. ACM* 40:3 pp. 77–87, 1997.
11. Maes, P., and M.S. Sharadanand, "Social information filtering: algorithms for automating "Word of Mouth"", *CHI Proc.* 1995.
12. Papadimitriou, C., P. Raghavan, H. Tamaki and S. Vempala, "Latent Semantic Indexing: A Probabilistic Analysis", *J. Comput. Systems Sciences*, 61. pp. 217-235, 2000.
13. V.V. Petrov, *Sums of Independent Random Variables*, Springer-Verlag, 1975.
14. Sheath, B. and P. Maes, "Evolving agents for personalised information filtering", *Proc. 9th IEEE Conf. on Artificial Intelligence for Applications*, 1993.
15. `http://www.sims.berkeley.edu/resources/collab/`.
16. Ungar L.H. and D.P. Foster, "A Formal Statistical Approach to Collaborative Filtering", *Conference on Automated Learning and Discovery* (CONALD), 1998.

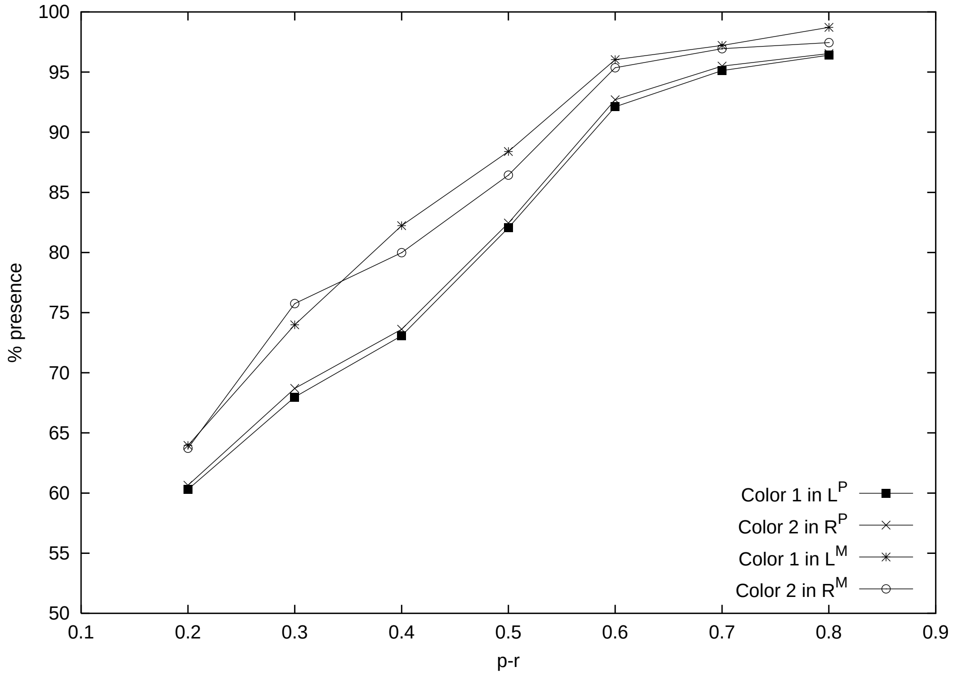

Fig. 1. Accuracy versus $p - r$.

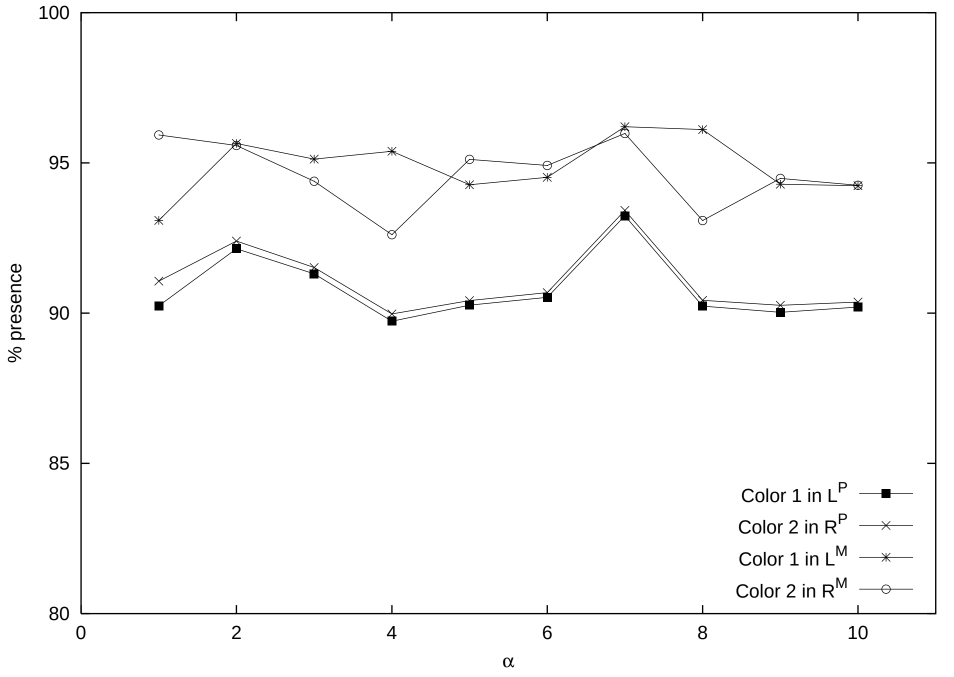

Fig. 2. Accuracy versus relative sizes.

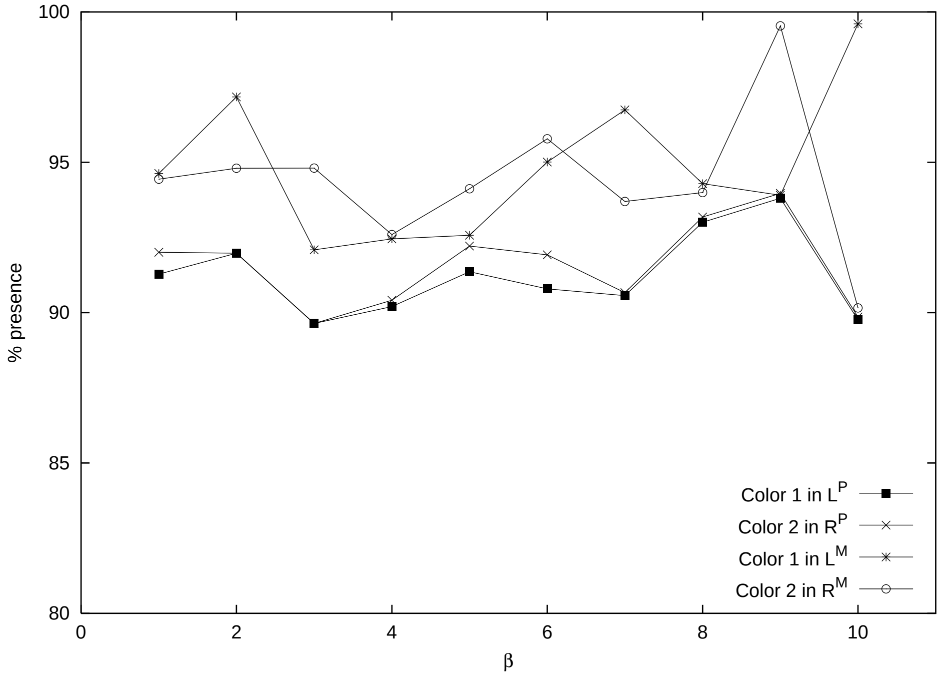

Fig. 3. Scalability. Here the size of the people and movie sets are $n = \beta n^*$ and $m = \beta m^*$, and n^*,m^* fixed.

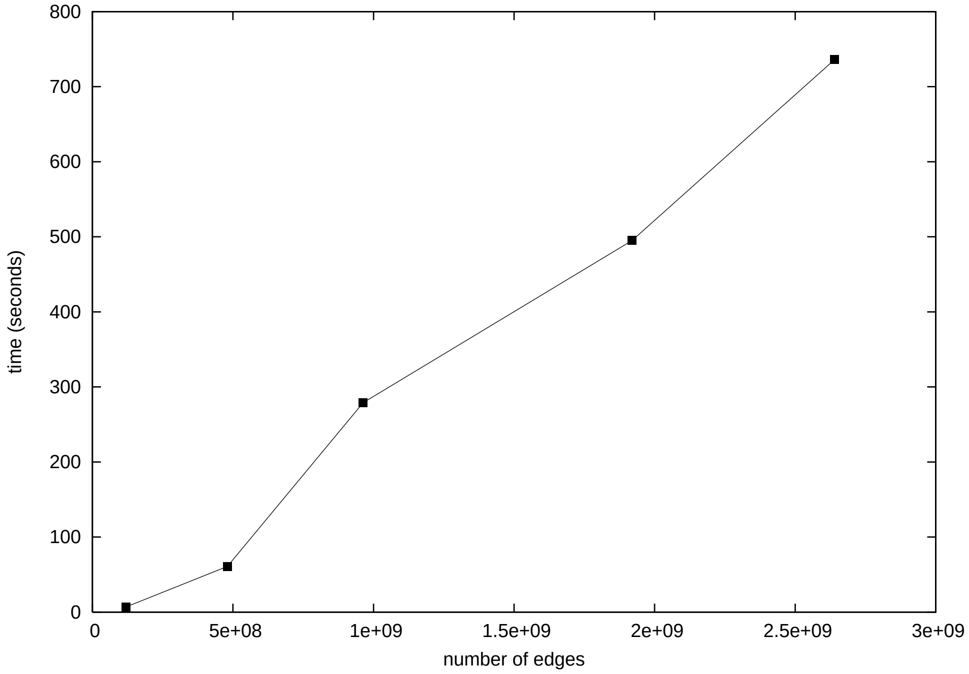

Fig. 4. Time performance of the semi-external memory algorithm.