Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence

Improved Local Search for Binary Matrix Factorization

Seyed Hamid Mirisaee
Univ. Grenoble Alps/CNRS
Grenoble, France
Hamid.Mirisaee@imag.fr

Eric Gaussier
Univ. Grenoble Alps/CNRS
Grenoble, France
Eric.Gaussier@imag.fr

Alexandre Termier
Univ. Rennes I/CNRS
Rennes, France*
Alexandre.Termier@irisa.fr

Abstract

Rank K Binary Matrix Factorization (BMF) approximates a binary matrix by the product of two binary matrices of lower rank, K, using either L_1 or L_2 norm. In this paper, we first show that the BMF with L_2 norm can be reformulated as an Unconstrained Binary Quadratic Programming (UBQP) problem. We then review several local search strategies that can be used to improve the BMF solutions obtained by previously proposed methods, before introducing a new local search dedicated to the BMF problem. We show in particular that the proposed solution is in general faster than the previously proposed ones. We then assess its behavior on several collections and methods and show that it significantly improves methods targeting the L_2 norms on all the datasets considered; for the L_1 norm, the improvement is also significant for real, structured datasets and for the BMF problem without the binary reconstruction constraint.

1 Introduction

Many datasets in scientific or industrial applications are high dimensional binary matrices. This high dimensionality negatively impacts the performance of traditional data analysis algorithms. Matrix factorization is a way to compress the data while preserving most of the characteristic patterns found inside, and has been a popular tool for dictionary learning, feature learning and subspace learning. When the input matrix and the factors are both binary, the operation is called a Binary Matrix Factorization (BMF). BMF has been successfully applied to many different contexts (Zhang et al. 2010; 2007; Meedsa et al. 2006; Li 2005; Zhang et al. 2007). When the problems considered involve more than several thousand elements, efficient strategies for finding better solutions to the matrix factorization problem in general, and the BMF one in particular, become interesting and challenging. Local search heuristics, as *p-opt* local search, can be used to improve the solution, corresponding to a local minimum of the objective function, provided by standard matrix factorization methods. However, such heuristics may be slow and any method that can speed them up would be interesting. This is exactly the objective of the present study for the BMF problem.

The general problem of rank K BMF takes the following form, given $\mathbf{X} \in \{0,1\}^{M\times N}$ and $K \in \mathbb{N}$, $K << \min(M,N)$:

$$\begin{cases} \underset{\mathbf{W},\mathbf{H}}{\text{argmin}} & ||\mathbf{X} - \mathbf{W}\times\mathbf{H}||_p \ (p=1 \text{ or } p=2) \\ \text{subject to} & \mathbf{W}\in\{0,1\}^{M\times K}, \mathbf{H}\in\{0,1\}^{K\times N} \\ & \text{(optional) } \mathbf{W}\times\mathbf{H}\in\{0,1\}^{M\times N} \end{cases} \quad (1)$$

where $||.||_p$ denotes either the L_1 norm ($p=1$) or the L_2 norm ($p=2$). The optional constraint is here referred to as the *binary reconstruction constraint*.

In this study, we show that BMF problem can be reformulated as a UBQP problem (Beasley 1998) when the L_2 norm is considered in Eq. (1). We explore this direction and propose a new *p-opt* local search procedure that we prove to be more efficient for BMF than the UBQP-based approach. The proposed *p-opt* can be applied to both the L_1 or L_2 methods. With thorough experiments on both synthetic and real data, we demonstrate that the heuristic we propose significantly improves the quality of existing methods in a reasonable amount of time. We also show that our local search procedure performs well when compared to other heuristic strategies (Kanungo et al. 2002).

2 Related work

The optimization problem (1) can be relaxed as an NMF (Lee and Seung 1999; 2000) or a SVD problem with additional binary constraints. One way to do that is to first solve a standard NMF or SVD problem and then to project the solution onto the $\{0,1\}$ sub-space (Miettinen et al. 2008). The second approach amounts to integrating the binary constraints into the objective function being minimized in NMF (Zhang et al. 2007; 2010). However, this latter approach does not generally yield a good approximation since increasing the coefficients of the regularization terms (which imposes the binary constraint) will result in high reconstruction error.

A simple and efficient heuristics to solve problem (1) is *Proximus* (Koyuturk, Grama, and Ramakrsihnan 2005). In this approach, one iteratively solves rank 1 decompositions

*Work done at Univ. Grenoble Alps/CNRS, Grenoble, France and combines them to form the global solution. In (Shen, Ji, and Ye 2009), the rank 1 BMF problem is formulated as a 0-1 integer linear program (ILP) which can be solved in a reasonable amount of time and provides initial solution to Proximus. These two latter studies are suitable for both L_1 and L_2 norm minimizations and guarantee all the conditions in (1). The study recently presented in (Jiang et al. 2014) shows an original approach for the L_1 norm minimization. In this work, the complete problem (with the binary reconstruction constraint) is addressed via $K-$means clustering. We will go back to this approach in Section 3.2.

The NMF problem as well as the $K-$means clustering one are known to be NP-hard (Gillis and Glineur 2008; Aloise et al. 2009). A related problem to the one considered here, namely the weighted rank 1 BMF, is also shown to be NP-complete in (Lu et al. 2011). Problem (1) has been related to different NP-hard problems and is in general conjectured to be NP-hard (as done in *e.g.* (Shen, Ji, and Ye 2009)) even though we know of no formal proof for this fact (neither a reduction from a known NP-hard problem nor a polynomial time algorithm have been provided so far, to the best of our knowledge).

3 General considerations

We establish in this section a relation between the BMF problem with L_2 norm and the Unconstrained Binary Quadratic Programming (UBQP) problem. This relation allows one to use the local search procedures designed for UBQP (Beasley 1998) to improve BMF solutions.

3.1 L_2-BMF and UBQP

Standard methods for L_2-BMF fix one matrix, $\mathbf{W}$ or $\mathbf{H}$, and solve for the other. The quantity to be minimized in L_2-BMF can be rewritten as:

$$||\mathbf{X} - \mathbf{WH}||_2^2 = \sum_{i=1}^{M}\sum_{j=1}^{N}(x_{ij} - \sum_{k=1}^{K} w_{ik}h_{kj})^2$$

with:

$$(x_{ij} - \sum_{k=1}^{K} w_{ik}h_{kj})^2 = x_{ij}^2 + (\sum_{k=1}^{K} w_{ik}h_{kj})^2 - 2x_{ij}\sum_{k=1}^{K} w_{ik}h_{kj}$$

Fixing e.g. $\mathbf{H}$ and solving for $\mathbf{W}$ thus amounts to solving the following problem, $\forall i,\ 1 \leq i \leq M$ (*i.e.* for all rows of $\mathbf{W}$):

$$\operatorname{argmin}_{\mathbf{w}_{i.}} \sum_{j=1}^{N}(\sum_{k=1}^{K} w_{ik}h_{kj})^2 + \sum_{k=1}^{K} w_{ik}\sum_{j=1}^{N}(-2x_{ij}h_{kj}) \tag{2}$$

where $\mathbf{w}_{i.}$ is the i^{th} row vector of $\mathbf{W}$. The first term in (2), $\sum_{j=1}^{N}(\sum_{k=1}^{K} w_{ik}h_{kj})^2$, can be rewritten as:

$$\sum_{k=1}^{K} w_{ik}\sum_{j=1}^{N} h_{kj} + \sum_{k=1}^{K}\sum_{k'=k+1}^{K} w_{ik}w_{ik'}2\sum_{j=1}^{N} h_{kj}h_{k'j}$$

Now, let us consider the symmetric, $K \times K$ matrix $\mathbf{Q}$:

$$q_{kk} = \sum_{j=1}^{N} h_{kj}(1-2x_{ij}),\ q_{kk'} = \sum_{j=1}^{N} h_{kj}h_{k'j}\ (k \neq k')$$

The quantity minimized in problem (2) can be rewritten with $\mathbf{Q}$ as follows:

$$\sum_{k=1}^{K} w_{ik}^2 q_{kk} + \sum_{k=1}^{K}\sum_{k'=k+1}^{K} 2w_{ik}w_{ik'}q_{kk'} = \mathbf{w}_{i.}^T\mathbf{Q}\mathbf{w}_{i.}$$

and thus problem (2) can be reformulated as:

$$\operatorname{argmin}_{\mathbf{w}_{i.}} \mathbf{w}_{i.}^T\mathbf{Q}\mathbf{w}_{i.} \tag{3}$$

which corresponds to a UBQP problem (Beasley 1998; Merz and Freisleben 2002). Applying the same development to $\mathbf{H}$, when $\mathbf{W}$ is fixed, leads to the following property.

Property 1. *Iteratively solving the L_2-BMF problem by fixing either $\mathbf{W}$ or $\mathbf{H}$ and solving for the other is equivalent to iteratively solving UBQP problems of the form:*

$$\operatorname{argmin}_{\mathbf{v}} \mathbf{v}^T\mathbf{Q}\mathbf{v}$$

with:

$$q_{kk} = \sum_{j=1}^{N} h_{kj}(1-2x_{ij}),\ q_{kk'} = \sum_{j=1}^{N} h_{kj}h_{k'j}\ (\mathbf{W}\textit{ fixed})$$

$$q_{kk} = \sum_{i=1}^{M} w_{ik}(1-2x_{ij}),\ q_{kk'} = \sum_{i=1}^{M} w_{ik}w_{ik'}\ (\mathbf{H}\textit{ fixed})$$

According to this formulation, it is interesting to improve L_2 norm solutions through local search algorithms designed for UBQP. In the context of BMF, the neighborhood of size p of a given $\mathbf{W}$ (or $\mathbf{H}$) is defined by the set of matrices that can be obtained from $\mathbf{W}$ (or $\mathbf{H}$) by flipping p cells. (Beasley 1998) and (Merz and Freisleben 2002) present such *p-opt* local search heuristics for UBQP. The *1-opt* local search algorithm proposed in (Merz and Freisleben 2002) looks for the solution in the neighborhood of size 1 that maximizes the gain with respect to the current solution and adopts this solution if the gain is positive. The *p-opt* local search (which parallels the Tabu search of (Glover F. 1998) discussed in (Beasley 1998) and is based on (Kernighan and Lin 1970) and (Lin and Kernighan 1973)) is similar for the neighborhood of size p, except that, for computational reasons, one does not look for the solution that maximizes the gain but only explores a part of the neighborhood looking for a solution that improves the current one. This exploration corresponds to a recursive application of the *1-opt* solution.

In general, *p-opt* local search algorithms are of course only interesting if the gains can be computed efficiently; the complexity of the *1-opt* local search algorithm (Merz and Freisleben 2002), when applied to the i^{th} row of $\mathbf{W}$ (problem 3) is $O(K^2)$ once the matrix $\mathbf{Q}$, which depends on i, has been constructed (the limiting factor here is the initial computation of the gains, which can be updated in $O(K)$ operations). The construction of $\mathbf{Q}$ has complexity $O(K^2N)$ (here also $\mathbf{Q}$ needs only be computed for the first row; it can then be updated for each row in $O(KN)$ operations). As a result, the overall complexity of *1-opt* for the UBQP problem is $O(K^2MN)$, which may be too high for practical applications when K is large (the complexity is the same when a column of $\mathbf{H}$ is considered).

In the remainder of the paper, we will refer to the *1-opt* local search proposed for UBQP as *1-opt-UBQP*.

3.2 L_1-BMF and K-means

In (Jiang et al. 2014), a $K-$means-based approach for the rank K BMF problem with L_1 norm minimization has been introduced. In this approach, first, K initial cluster centers are selected from column vectors of $\mathbf{X}$; then, each column vector of $\mathbf{X}$ is assigned to the class of the closest center. New binary center are then computed, and the process is repeated until the clusters do not evolve. This approach, referred to as *CBMF*, solves the rank K binary factorization problem with the binary reconstruction constraint. A slight modification of this method, referred to as *UBMF*, aims at solving the problem without the binary reconstruction constraint.

Because of its reliance on K-means, CBMF can be improved through local search procedures designed for K-means. In (Kanungo et al. 2002), such procedure, called *swap procedure*, is introduced and that consists, when applied to our problem, in randomly selecting one of the final centers of the current CBMF solution and replace it with one random column of $\mathbf{X}$. The K-means algorithm is then applied in order to see if the new solution is better than the previous one; if it is the case, it becomes the current solution. This process is repeated a certain number of times or until the time budget has been exhausted.

4 Efficient *p-opt* local search for BMF

As illustrated in Eq. 2, rows of $\mathbf{W}$ (or columns of $\mathbf{H}$) can be optimized independently. Here, we show how to optimize, in a neighborhood of size 1, a row of $\mathbf{W}$ (the reasoning is the same for columns of $\mathbf{H}$). We first divide each row of $\mathbf{W}$, $\mathbf{w}_{i.}$, into three sets as follows.

Definition 1. *For a given row $\mathbf{w}_{i.}$ $(1 \leq i \leq M)$ of $\mathbf{W}$ and a given $\mathbf{H}$, we define three sets of column vectors of $\mathbf{H}$:*

$$W_i^0 = \{\mathbf{h}_{.l} \mid 1 \leq l \leq N,\ x_{il} = 0\}$$

$$W_i^{\perp} = \{\mathbf{h}_{.l} \mid 1 \leq l \leq N,\ x_{il} = 1,\ <\mathbf{w}_{i.}, \mathbf{h}_{.l}>= 0\}$$

$$W_i^{\not\perp} = \{\mathbf{h}_{.l} \mid 1 \leq l \leq N,\ x_{il} = 1,\ <\mathbf{w}_{i.}, \mathbf{h}_{.l}>\neq 0\}$$

where $\mathbf{h}_{.l}$ represents the l^{th} column of $\mathbf{H}$, x_{il} denotes the cell of $\mathbf{X}$ at row i and column l and $<.,.>$ is the dot product. We have of course: $|W_i^0| + |W_i^{\perp}| + |W_i^{\not\perp}| = N$ (number of columns in $\mathbf{H}$). The rationale for considering the three sets above comes from the fact that the number of mistakes in the i^{th} row of the reconstructed matrix $\mathbf{W} \times \mathbf{H}$ can be easily computed from them, as shown in Theorems 1 and 2.

Let $\Delta E(i,j) = E(i,j)_{\text{new}} - E(i,j)_{\text{old}} = ||\mathbf{x}_{i.} - \mathbf{w}_{i.}^j \mathbf{H}||_p - ||\mathbf{x}_{i.} - \mathbf{w}_{i.} \times \mathbf{H}||_p$, where $\mathbf{w}_{i.}^j$ is obtained from $\mathbf{w}_{i.}$ by flipping the j^{th} bit (from 0 to 1 or from 1 to 0), denote the difference in the reconstruction error before and after a bit flip. Theorem 1 and 2 provide simple expressions for $\Delta E(i,j)$ for both L_1 and L_2 norms. These expressions can be computed efficiently as shown in Theorem 3.

Theorem 1. *Let $\Delta E(i,j; 0 \to 1, L_2)$ be the gain obtained when flipping the j^{th} bit of $\mathbf{w}_{i.}$ from 0 to 1 (resp. from 1 to 0), measured by the L_2 norm. Then:*

$$\Delta E(i,j;0 \to 1, L_2) = \sum_{\mathbf{h} \in W_i^0} (1 + 2 <\mathbf{w}_{i.}, \mathbf{h}>) h_j + \sum_{\mathbf{h} \in W_i^{\not\perp}} (2 <\mathbf{w}_{i.}, \mathbf{h}> -1) h_j - \sum_{\mathbf{h} \in W_i^{\perp}} h_j \tag{4}$$

and:

$$\Delta E(i,j;1 \to 0, L_2) = \sum_{\mathbf{h} \in W_i^0} (1 - 2 <\mathbf{w}_{i.}, \mathbf{h}>) h_j + \sum_{\substack{\mathbf{h} \in W_i^{\not\perp} \\ <\mathbf{w}_{i.},\mathbf{h}> h_j = 1}} 1 + \sum_{\substack{\mathbf{h} \in W_i^{\not\perp} \\ <\mathbf{w}_{i.},\mathbf{h}> h_j > 1}} 3 - 2 <\mathbf{w}_{i.}, \mathbf{h}> \tag{5}$$

Theorem 2. *Similar to Theorem 1, for L_1 we have the following gain computations:*

$$\Delta E(i,j;0 \to 1, L_1) = \sum_{\mathbf{h} \in W_i^0} h_j + \sum_{\mathbf{h} \in W_i^{\not\perp}} h_j - \sum_{\mathbf{h} \in W_i^{\perp}} h_j \tag{6}$$

$$\Delta E(i,j;1 \to 0, L_1) = \sum_{\substack{\mathbf{h} \in W_i^{\not\perp} \\ <\mathbf{w}_{i.},\mathbf{h}> h_j = 1}} 1 - \sum_{\mathbf{h} \in W_i^0} h_j - \sum_{\substack{\mathbf{h} \in W_i^{\not\perp} \\ <\mathbf{w}_{i.},\mathbf{h}> h_j > 1}} 1 \tag{7}$$

We omit the proofs, which are mainly technical, of these theorems for space reasons.

As mentioned in the previous section, the complexity for updating one row (say row i) of $\mathbf{W}$ through *1-opt-UBQP* (*i.e.* the *1-opt* procedure defined for UBQP and explained in (Merz and Freisleben 2002)) is $O(K^2 N)$. A standard procedure, *i.e.* computing the gain through the multiplication of $\mathbf{w}_{i.}$ and $\mathbf{H}$, would in fact have the same complexity (as this multiplication can be done in $O(KN)$ and needs to be done K times). We show here that the *1-opt* procedure defined on the basis of Theorem 1 and 2, and referred to as *1-opt-BMF* is more efficient.

Theorem 3. *We assume a row vector $\mathbf{w}_{i.}$ of $\mathbf{W}$ and a matrix $\mathbf{H}$. Furthermore, let d be the density of $\mathbf{X}$ and let τ denote the proportion of columns l of $\mathbf{H}$ orthogonal to $\mathbf{w}_{i.}$ and such that $x_{il} = 1$ (thus $\tau = |W_i^{\perp}|/N$). Then, the gain in complexity for the 1-opt-BMF procedure based on Theorem 1 and 2 compared to both 1-opt-UBQP and the standard approach is at least:*

- $((1 - (d - \tau)) + \frac{d-\tau}{K})^{-1}$ *for the L_2 norm;*
- $\min\{((d - \tau) + \frac{1-d}{K})^{-1}, K\}$ *for the L_1 norm.*

Proof (sketch) First note that $0 \leq \tau \leq d \leq 1$ as the proportion of columns in $\mathbf{H}$ corresponding to 1s in $\mathbf{x}_{i.}$, *i.e.* d, is larger than the proportion of columns in $\mathbf{H}$ corresponding to 1s in $\mathbf{x}_{i.}$ and orthogonal to $\mathbf{w}_{i.}$, *i.e.* τ. From the definitions of d and τ, one has: $|W_i^0| = (1-d)N,\ |W_i^{\perp}| = \tau N,\ |W_i^{\not\perp}| = (d-\tau)N$. The complexity of computing Eq. 4 thus amounts to: $K(1-d)N+K\tau N+(d-\tau)N = KN(1-(d-\tau)+(d-\tau)/K)$ as the first two terms of Eq. 4 involve a dot product with $\mathbf{w}_{i.}$ whereas the last one does not. Similarly, the complexity of Eq. 5 is $K(1-d)N+K(d-\tau)N = KN(1-\tau)$, which is lower than the previous one. For *1-opt* UBQP (and for the naive method), the complexity for the same operation is KN. Dividing this latter quantity with the first one yields the first part of Theorem 3. The second part is obtained in a similar way, except that this time the first quantity is not always larger than the second; it depends on the position of K wrt $d/d-\tau$. □

For $K = 1$, the local search procedure based on Theorem 1 and 2 has the same complexity (disregarding multiplicative and additive constants) as the standard ones; when K increases, provided that $d-\tau$ is fixed, the advantage of the local search procedure based on Theorem 1 and 2 with respect to both *1-opt-UBQP* and the standard approach should be more obvious. This advantage is however less clear when τ is close to d. The results obtained in the experimental section confirm these points.

This procedure can naturally be extended to *p-opt* local search, either by adopting a *greedy* approach, in which one applies p times a *1-opt* local search, as done in (Merz and Freisleben 2002), or by finding the p flips that maximize the gain. In both cases, the gain of flipping several bits is the sum of the gain of the individual flips. However, in the latter approach, the complexity for selecting the best bits to flip is in $O(K^p)$, which could be very time consuming in most practical cases (in contrast, the complexity of the *p-opt* greedy approach to select the p bits is $O(pK)$). Lastly, one should note that the binary reconstruction condition can be efficiently checked while computing the gains defined in Theorem 1 and 2.

5 Experiments

5.1 General Settings and Datasets

We evaluated the proposed *1-opt-BMF* procedure on different methods and on different datasets. We have retained here the most widely used methods to solve the BMF problem: projected NMF, projected SVD, *Proximus*, CBMF and UBMF. All the implementations have been done in Matlab which offers efficient matrix operations. For SVD and NMF, we used the Matlab built-in functions and for the rest we have used our own efficient implementations (all the codes are available from the authors upon request).

For projected NMF and SVD, we found the best projection points by applying a grid search, with a step size of 0.05. For efficiency reasons, we mainly focus on $p = 1$ in the *p-opt* local search.

As mentioned before, we refer to the method proposed in this paper as *1-opt-BMF*, to the standard implementation as *1-opt-Standard* and to the *1-opt* local search associated with UBQP as *1-opt-UBQP*. One should note that the *1-opt-Standard* approach benefits from highly efficient block algorithms available in LAPACK (incorporated in Matlab). In average, these algorithms are significantly faster than a simple, naive multiplication.

In addition, we evaluate the swap heuristic described in (Kanungo et al. 2002) that can be applied over CBMF results (we call it *SCBMF* where S stands for "swap"). We also consider applying *1-opt-BMF* over SCBMF in order to see if it can further improve the solution. This last approach is referred to as *1-opt-SCBMF*. There are different criteria to choose the number of swaps in SCBMF. For example, one can select a fixed number of swaps or a time limit. To be fair in our comparison, we have adopted here the following strategy: for each dataset we first run our *1-opt* algorithm (which is in general faster) on CBMF and measure its running time and let the SCBMF procedure run the same amount of time to improve CBMF results. Like that, one can observe which method performs best under a given time budget.

The main difference between SCBMF and all *1-opt* local search procedures is that the former tries to find better solutions by exploring different parts of the search space whereas the latter tries to find a better solution in a close neighborhood of the current solution. Observing the respective behaviors of these different approaches is thus interesting. Lastly, since SCBMF proceeds via random selections, we run the SCBMF 10 times for each dataset and report the average error. In this case, a difference is deemed statistically significant if it was significant on the majority of the 10 runs. We use here the Wilcoxon sum-rank test at 1% significance level to assess whether differences in the results are significant or not.

We let the rank K vary in the set $\{1, 20, 40, 60, 80, 100\}$. As mentioned above, the BMF problem can be solved with or without the binary reconstruction constraint, with the L_1 norm or with the L_2 norm. Accordingly, four classes of decomposition methods can be considered (Table 1). Following the methodology used in (Uno, Kiyomi, and Arimura 2005), we examined both real world datasets and synthetic ones, all available at *http://fimi.ua.ac.be/data/* (last visited 15-Nov-2014). Table 2 shows the sets we used in addition to some of their characteristics. The first part of the table shows the real datasets and the second part shows the synthetic ones, generated by the IBM Data Generator. We also removed all empty rows and columns from the data.

Table 1: Different problem classes

Approach/Norm	L_1	L_2
Constrained	CBMF	PROXIMUS
Unconstrained	UBMF	NMF, SVD

Table 2: Datasets used in the experiments

Name	**# Rows**	**# Columns**	**Type**
Mushroom	8124	119	sparse
Accidents	340183	468	dense
Connect	67557	129	structured dense
Pumsb	49046	2113	structured dense
T10I4D100K	100000	870	very sparse
T40I10D100K	100000	942	sparse

5.2 Time Comparison

We compare here the proposed *1-opt-BMF* with *1-opt-Standard* and *1-opt-UBQP* according to their running time

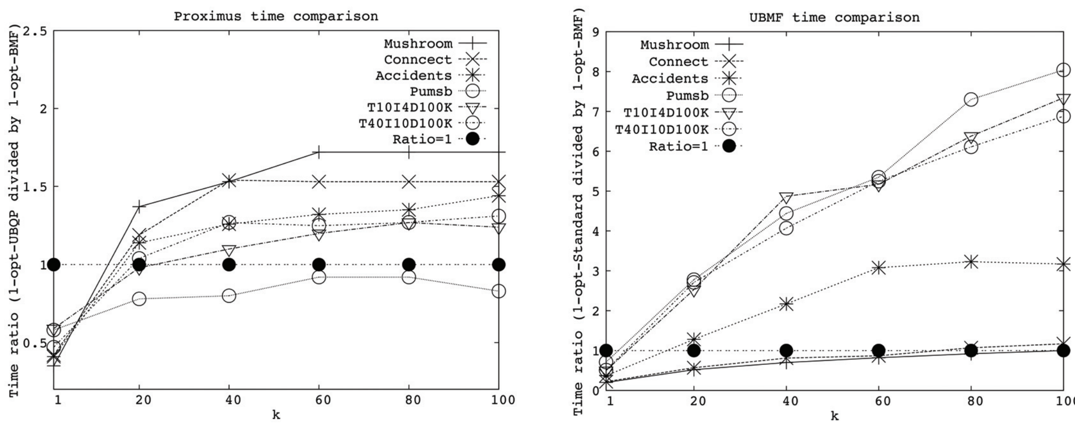

Figure 1: Time ratio for *Proximus* (left) and UBMF (right)

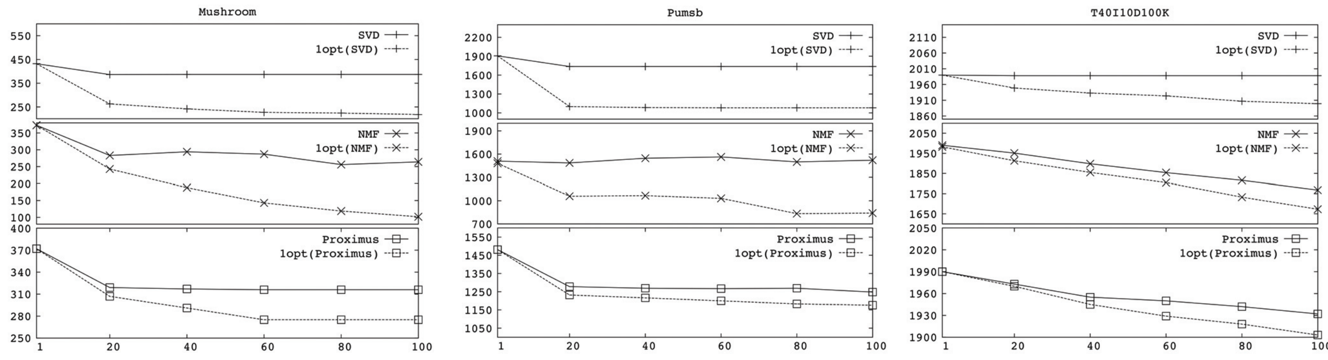

Figure 2: Impact of *1-opt-BMF* on the L_2 norm methods (horizontal and vertical axes show values of *K* and L_2 norm resp.)

(one should note that these three approaches yield the same solution, namely the one corresponding to the best improvement in a neighborhood of size 1). Note that *1-opt-BMF* and *1-opt-Standard* can be applied to both L_1 and L_2 decomposition methods, whereas *1-opt-UBQP* can only be applied on the L_2 decomposition approaches (NMF, SVD and *Proximus*).

To further illustrate the speed of the different methods, we display in Figure 1 the **ratio**: execution time of *1-opt-Standard* divided by execution time of *1-opt-BMF* or execution time of *1-opt-UBQP* divided by execution time of *1-opt-BMF*. For space reasons, we only show one of the L_1 norm methods (UBMF) and one of the L_2 norm methods (*Proximus*), but the figures of other decomposition methods are very similar to the ones shown here. An additional line, labeled "Ratio=1", is added to make the comparison easier: the proposed *1-opt* (*i.e. 1-opt-BMF*) is faster when the curve is above this line, and slower otherwise.

As one can see, on 4 out of 6 datasets, *1-opt-BMF* is significantly faster than *1-opt-Standard* (Figure 1, right). The two collections on which this is not observed are `Mushrooms` and `Connect`, which are the two smallest collections considered here. The *1-opt-Standard* procedure does not need to compute the different sets used by *1-opt-BMF*, and is more efficient when the number of columns, N, is small. However, as N increases, *1-opt-BMF* becomes faster. Furthermore, the difference increases in favor of *1-opt-BMF* when K increases which is in line with the theoretical analysis provided in section 4. For sufficiently large datasets and sufficiently large values of K, the proposed *1-opt-BMF* can be up to 8 times faster than the standard approach.

1-opt-BMF is also faster on 5 datasets out of 6 (Figure 1, left) comparing to *1-opt-UBQP*. Here again the advantage of *1-opt-BMF* roughly increases with K. However, this advantage is not as marked as it is for *1-opt-Standard*, which shows that *1-opt-UBQP* is indeed more efficient than a standard approach for improving a given solution.

By definition, all *1-opt* approaches yield the same results as they explore exactly the same space; the only difference between them lies in the efficiency with which the neighborhood is explored. In the remainder of the paper, we focus on *1-opt-BMF*, the most efficient *1-opt* local search procedure. Lastly, 1-opt-BMF and 1-opt-standard have the same space complexity and the same access patterns (and thus have similar locality properties), while the space complexity of 1-opt-UBQP is larger as it requires to keep matrix $\mathbf{Q}$ in the memory.

5.3 Impact of *1-opt-BMF* for L_2-BMF

Figure 2 shows the effectiveness of the *1-opt-BMF* strategy when it is applied on (projected) SVD, (projected) NMF and

Table 3: Effectiveness of the *1-opt-BMF* approach on L_1 methods

Dataset	k	CBMF		SCBMF	*1-opt-SCBMF*	UBMF	
		Standard	*1-opt-BMF*			Standard	*1-opt-BMF*
Mushroom	20	7.5032	**7.2662**	7.0481	**6.7332**	6.8685	6.8685
	40	4.9601	**4.4301**	**4.1346**	**3.6278**	4.4537	4.4537
	60	2.5769	**2.2591**	2.4711	**2.1682**	2.1958	2.1958
	80	1.4323	**1.1655**	1.3449	**1.1608**	1.0042	1.0042
	100	0.1917	0.1313	0.1835	**0.1500**	0.1377	0.1377
Pumsb	20	1.3253	**1.2895**	1.3214	**1.2782**	1.2177	1.2176
	40	0.9608	**0.9310**	0.9608	**0.9311**	0.8903	0.8903
	60	0.8719	**0.8647**	0.8719	**0.8647**	0.8201	0.8200
	80	0.8763	**0.8467**	0.8740	**0.8491**	0.7638	0.7637
	100	0.7560	**0.7470**	0.7560	**0.7470**	0.7010	0.7010
Connect	20	7.9585	**7.8659**	**7.4513**	**7.3004**	7.3202	7.3202
	40	4.9624	**4.9077**	**4.5270**	**4.4728**	4.1145	4.1145
	60	2.6073	2.5850	**2.4071**	2.3849	1.7336	1.7336
	80	1.7881	**1.6420**	**1.3584**	**1.3551**	0.8995	0.8995
	100	0.3537	0.3537	0.3537	0.3537	0.3537	**0.3321**

Proximus. For space reasons, we only illustrate 3 datasets here; the results are similar for other datasets. As one can note, projected NMF performs generally better than projected SVD, with a reconstruction error significantly lower. Projected NMF also yields a lower reconstruction error than *Proximus*, but *Proximus* provides a binary reconstructed matrix, which is not the case for NMF.

To assess whether the improvements obtained by the *1-opt-BMF* are significant, we computed *p-value* of the Wilcoxon rank-sum test at the 1% significance level. In almost all L_2 norm cases with $K = 1$ there is no improvement with *1-opt-BMF*. This can be explained by the fact that the first dominant factor in the datasets is more easily approximated by the different methods than the first K factors. For $K > 1$ however, in all sets and with all methods, the improvements obtained by the *1-opt-BMF* are statistically significant. In particular, for NMF and SVD, the improvement can be up to 61% for real sets (as `Pumsb`), and up to 12 % for synthetic sets (as `T1014D100K`).

5.4 Impact of *1-opt* for L_1-BMF

Table 3 illustrates the improvement that *1-opt-BMF* can make over the L_1 methods, namely CBMF, SCBMF and UBMF. As one can note, the case of $K = 1$ is removed from the table since, as mentioned above, no local improvement can be made for the first factor. We only show three real datasets; the results for other real sets are similar to the ones shown in the table. We skipped the synthetic data since, for all methods, there is not much improvement with local heuristics according to the lack of structure in these datasets. The table shows the normalized errors in percentage (normalized to the total number of cells). Significant improvements are shown in bold (measured again using Wilcoxon rank-sum test at the 1% significance level). Note that for CBMF (resp. UBMF), the *1-opt-BMF* error is shown in bold if it is significantly better than the standard CBMF (resp. UBMF). SCBMF's error is shown in bold if it is significantly better than standard CBMF. For *1-opt-SCBMF*, the error is in bold if it is significantly better than SCBMF.

The *1-opt-BMF* strategy consistently yields significant improvements over the standard CBMF solutions. The most spectacular case is the `Mushroom` dataset, where for $K = 80$ the error value is decreased by more than 18%. This dataset has a large number of patterns spanning few lines (Uno, Kiyomi, and Arimura 2005), making it more difficult for factorization algorithms to compute a good approximation for high K values. These results show that the *1-opt-BMF* heuristic can help improve the decomposition on such difficult cases. It is also interesting to see that SCBMF can only bring one significant improvement for this dataset while the *1-opt-BMF* can make four significant improvements (out of five cases).

Although `Pumsb` is a *structured* dense dataset, it contains very few one cells (only 3.5%). This issue, combined with the higher number of columns, prevents SCBMF to find a better solution in the allocated time. However, despite this difficulty, *1-opt-BMF* significantly improves the solution found by CBMF. (at best 3% for $K = 40$).

The conclusions for UBMF are more contrasted, as it is more difficult to improve the results in this case. This can be explained by the fact that UBMF solves the BMF problem without the binary reconstruction constraint, and thus yields solutions with lower reconstruction errors. Nevertheless, there are still a few cases where the *1-opt-BMF* approach can provide a significantly better approximation, as `Connect` with $K = 100$.

6 Conclusion

We have first shown that the BMF problem based on L_2 norm reconstruction can be reformulated as a UBQP problem, prior to review several local search strategies that can be used to improve BMF solutions (for both L_1 and L_2 norm reconstruction) obtained by factorization methods. We have then introduced a new local search method and studied its complexity with respect to other local search approaches.

Our experiments, conducted with several state-of-the-art methods, on several collections with different properties, have confirmed that the proposed *1-opt-BMF* procedure is in general faster than the previously proposed ones. We also have shown that, given a current solution obtained by any matrix factorization method, the *1-opt-BMF* local search can find a significantly better solution in most cases.

References

Aloise, D.; Deshpande, A.; Hansen, P.; and Popat, P. 2009. Np-hardness of euclidean sum-of-squares clustering. *Machine Learning* 75(2).

Beasley, J. E. 1998. Heuristic algorithms for the unconstrained binary quadratic programming problem. *London, UK: Management School, Imperial College* 4.

Gillis, N., and Glineur, F. 2008. Nonnegative factorization and the maximum edge biclique problem. In *CORE Discusssion paper*.

Glover F., Kochenberger G. A., A. B. 1998. Adaptive memory Tabu search for binary quadratic programs. *Management Science* 44.

Jiang, P.; Peng, J.; Heath, M.; and Yang, R. 2014. A clustering approach to constrained binary matrix factorization. In *Data Mining and Knowledge Discovery for Big Data*. Springer. 281–303.

Kanungo, T.; Mount, D. M.; Netanyahu, N. S.; Piatko, C. D.; Silverman, R.; and Wu, A. Y. 2002. A local search approximation algorithm for k-means clustering. In *Proceedings of the eighteenth annual symposium on Computational geometry*, 10–18. ACM.

Kernighan, B., and Lin, S. 1970. An Efficient Heuristic Procedure for Partitioning Graphs. *The Bell Systems Technical Journal* 49(2).

Koyuturk, M.; Grama, A.; and Ramakrsihnan, N. 2005. Compression, clustering, and pattern discovery in very-high-dimensional discrete-attribute data sets. *IEEE Trans. Knowledge Data Eng* 17:447–461.

Lee, D. D., and Seung, H. S. 1999. Learning the parts of objects by non-negative matrix factorization. *Nature* 401(6755):788–791.

Lee, D. D., and Seung, H. S. 2000. Algorithms for non-negative matrix factorization. In *In NIPS*, 556–562. MIT Press.

Li, T. 2005. A general model for clustering binary data. *ACM SIGKDD* 188–197.

Lin, S., and Kernighan, B. W. 1973. An effective heuristic algorithm for the traveling-salesman problem. *Operations research* 21(2):498–516.

Lu, H.; Vaidya, J.; Atluri, V.; Shin, H.; and Jiang, L. 2011. Weighted rank-one binary matrix factorization. In *SDM*, 283–294.

Meedsa, E.; Ghahramani, Z.; Neal, R.; and Roweis, S. 2006. Modeling dyadic data with binary latent factors. *NIPS* 977–984.

Merz, P., and Freisleben, B. 2002. Greedy and local search heuristics for unconstrained binary quadratic programming. *Journal of Heuristics* 8(2):197–2013.

Miettinen, P.; Mielikäinen, T.; Gionis, A.; Das, G.; and Mannila, H. 2008. The discrete basis problem. *IEEE Trans. Knowl. Data Eng.* 20(10):1348–1362.

Shen, B.; Ji, S.; and Ye, J. 2009. Mining discrete patterns via binary matrix factorization. *ACM SIGKDD* 757–766.

Uno, T.; Kiyomi, M.; and Arimura, H. 2005. Lcm ver. 3: Collaboration of array, bitmap and prefix tree for frequent itemset mining. In *Proceedings of the 1st international workshop on open source data mining: frequent pattern mining implementations*, 77–86. ACM.

Zhang, Z.; Li, T.; Ding, C.; Ren, X.; and Zhang, X. 2007. Binary matrix factorization with applications. *ICDM* 391–400.

Zhang, Z.; Li, T.; Ding, C.; Ren, X. W.; and Zhang, X. 2010. Binary matrix factorization for analyzing gene expression data. *Data Mining and Knowledge Discovery* 20(1):28–52.