PUC

ISSN 0103-9741

Monografias em Ciência da Computação

nº 02/2023

Towards efficient searches for the Discrete Basis Problem

Georges Spyrides

Marcus Poggi

Hélio Lopes

Departamento de Informática

PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO

RUA MARQUÊS DE SÃO VICENTE, 225 - CEP 22451-900

RIO DE JANEIRO - BRASIL

Monografias em Ciência da Computação, No. 02/2023 ISSN: 0103-9741
Editor: Prof. Editor Principal November, 2020

Towards efficient searches for the Discrete Basis Problem

Georges Spyrides, Marcus Poggi and Hélio Lopes

gspyrides@inf.puc-rio.br, poggi@inf.puc-rio.br, lopes@inf.puc-rio.br

Abstract. The discrete basis problem is a variant of the set covering/partitioning problem, in which not covering is allowed, but penalized in the objective function, and over-covering an item is also penalized. This problem sparked new interest recently as a subproblem for algorithms applied to Binary Dictionary Learning, Frequent Itemset Discovery, and, most importantly, Binary Matrix Factorization (BMF). The BMF can be used for clustering items, categorical characteristics of observations, and recommendation systems for users interacting with itemsets. The most common algorithms approximate the factorization through gradient descent. We achieved theoretical results that greatly improve solving the discrete basis problem. These results will enable a backtracking approach that can solve the linearized formulation of the subproblem in large binary matrices taking advantage of their sparsity in real settings.

Keywords: binary matrix factorization, non-negative matrix factorization, integer programming, discrete basis problem

Resumo. O problema de base discreta é uma variante do problema de cobertura/ particionamento de conjuntos, no qual não cobrir um item não é proibido, mas é penalizada na função objetivo, e a sobrecobertura de um item também é penalizada. Este problema despertou recentemente um novo interesse como um subproblema para algoritmos aplicados ao Aprendizado de Dicionário Binário, à Descoberta Frequente de Conjuntos de Itens e, mais importante, à Fatoração de Matrizes Binárias (BMF). O BMF pode ser usado para agrupar itens, características categóricas de observações e sistemas de recomendação para usuários que interagem com conjuntos de itens. Os algoritmos mais comuns aproximam a fatoração por gradiente descendente. No entanto, os resultados são aproximadamente binários. Obtivemos resultados teóricos que melhoram a resolução do problema de bases discretas. Esses resultados permitirão uma abordagem de busca recursiva que pode resolver a formulação linearizada do subproblema em grandes matrizes binárias, ainda aproveitando sua esparsidade em situações reais.

Palavras-chave: fatoração binária de matrizes, fatoração não-negativa de matrizes, programação inteira, problema de bases discretas

In charge of publications:
PUC-Rio Departamento de Informática - Publicações
Rua Marquês de São Vicente, 225 - Gávea
22453-900 Rio de Janeiro RJ Brasil
Tel. +55 21 3527-1516 Fax: +55 21 3527-1530
E-mail: publicar@inf.puc-rio.br
Web site: http://bib-di.inf.puc-rio.br/techreports/

Table of Contents

1 Problem Formulation and the connection to binary matrix factorization

In the Binary Matrix Factorization Setting, the matrix $A \in \{0,1\}^{|\mathcal{M}|,|\mathcal{G}|}$ is given as input and an algorithm tries to find $W \in [0,1]^{|\mathcal{M}|,|\mathcal{G}|}$ and $H \in [0,1]^{|\mathcal{G}|,|\mathcal{N}|}$, such that the multiplication $W \cdot H$ is an approximation for A.

In some cases the entries of W and H are real values between 0 and 1, and for many applications, we want to discretize these entries without losing too much of the reconstruction error, which measures the distance of approximation between A and the reconstruction $W \cdot H$.

One common way to measure this approximation error is to measure how much of the variance was captured. Thus, we measure how close to zero is the difference $A - W \cdot H$. We can calculate this by measuring the norm of this difference relative to the norm of the original matrix A, as shown in equation 1.

$$\text{minimize } \|A - W \cdot H\|_2 = \sum_{m \in \mathcal{M}} \sum_{n \in \mathcal{N}} (a_{mn} - (\sum_{g \in \mathcal{G}} w_{mg} \cdot h_{gn}))^2 \tag{1}$$

Using this equation as an optimization problem we observe some characteristics that suggest hardness even for approximations: binary decision variables, bilinearity and quadratic objective. Therefore, a common approach is to try approximating the problem using a linearized surrogate objective function using the ℓ_1-norm, as shown in equation 2.

$$\text{minimize } \|A - W \cdot H\|_1 = \sum_{m \in \mathcal{M}} \sum_{n \in \mathcal{N}} |a_{mn} - (\sum_{g \in \mathcal{G}} w_{mg} \cdot h_{gn})| \tag{2}$$

Our approach relies on obtaining a first approximation to one of the matrices, preferably W first, and solving a subproblem problem of approximating each column of the given matrix A as a sum of columns of W, obtaining matrix H. Then, fixing the value of matrix H and solving the transposed view of the first step, approximating each row of A as a sum of a subset of rows of H, obtaining a new value for W. This subproblem is called by Miettinen in [2] as the Discrete Basis Problem.

Assuming a first approximation for W as fixed, a simple rearrangement of equation 2 shows that we can treat the summation over rows in $\mathcal{M}$ separately for each column in set $\mathcal{N}$.

$$\text{minimize } \sum_{m \in \mathcal{M}} \sum_{n \in \mathcal{N}} |a_{mn} - (\sum_{g \in \mathcal{G}} w_{mg} \cdot h_{gn})| = \sum_{n \in \mathcal{N}} \left[\text{minimize } \sum_{m \in \mathcal{M}} |a_{mn} - (\sum_{g \in \mathcal{G}} w_{mg} \cdot h_{gn})| \right] \tag{3}$$

Another way of thinking about this subproblem is a problem of choosing a subset of a binary basis to represent a given binary vector. We show this interpretation in equation 4. In this equation we have to approximate the column a_n using the binary decision variables h_{gn} to choose from a set of fixed basis, the columns of W.

$$\begin{bmatrix} | \\ a_n \\ | \end{bmatrix} \approxeq \begin{bmatrix} | \\ w_1 \\ | \end{bmatrix} h_{1n} + \begin{bmatrix} | \\ w_2 \\ | \end{bmatrix} h_{2n} + \cdots + \begin{bmatrix} | \\ w_g \\ | \end{bmatrix} h_{gn} + \cdots + \begin{bmatrix} | \\ w_{\mathcal{G}} \\ | \end{bmatrix} h_{\mathcal{G}n} \tag{4}$$

We can also run an similar procedure fixating H and optimizing matrix W, one row of A at a time, by just transposing the multiplication.

$$A_{[m\times n]} \approxeq W_{[m\times g]} \cdot H_{[g\times n]} \rightarrow A^T_{[n\times m]} \approxeq H^T_{[n\times g]} \cdot W^T_{[g\times m]} \tag{5}$$

Therefore, the discrete basis in this transposed view becomes a selection of rows of H to approximate each row t of matrix A.

$$\begin{bmatrix} | \\ a_m^T \\ | \end{bmatrix} \approxeq \begin{bmatrix} | \\ h_1^T \\ | \end{bmatrix} w_{m1} + \begin{bmatrix} | \\ h_2^T \\ | \end{bmatrix} w_{m2} + \cdots + \begin{bmatrix} | \\ h_g^T \\ | \end{bmatrix} w_{mg} + \cdots + \begin{bmatrix} | \\ h_{\mathcal{G}} \\ | \end{bmatrix} w_{m\mathcal{G}} \tag{6}$$

Consequently, a single algorithm for this subproblem can be used to optimally solve the linearized optimization formulation described in equation 3 looping through each row then through each column.

2 Related work

A pioneer work of [2] describes the ASSO algorithm. This algorithm uses pairwise distance between rows to decide which position should be rounded to one, managing to maximize coverage of the target matrix and minimize overlapping positions. In the same paper, the authors briefly present alternatives for the ASSO, including using k-means and exhaustive search.

Mirisaee et al in [3] propose a neighborhood for searching improvements in each row. Additionally, the authors presents different versions of the search by linearizing the objective function, which is an idea explored by us. Also, there is a work using genetic algorithms in order to search for solutions introduced by Snášel et al. in [5] and [4].

In a recent work Kovacs et al. [1] proposes a column generation approach, although they test with ranks up to 6. Which is telling of the hardness of the problem, normally a exact formulation would introduce a number of variables at least proportional to the product of the number of rows to the number of columns.

3 A change of perspective

Usually, in most examples, the set of basis that we must choose from are really sparse. We begin with an example of the discrete basis problem using the matrix notation, so that we can present the main intuition behind the core ideas of this paper.

Then, we define how to work the problem using set notation. The change to set representation allow us to both work with the reduced space of doing computations only proportional to the number of entries equal to one, but also to prove that an exhaustive search beginning from a trivial solution, can be efficient.

Finally, we revisit the same example under the lens of this theoretical framework.

3.1 Example

Suppose we are trying to approximate a column by choosing a combination of basis and minimizing the ℓ_1-norm of the difference between the original and the reconstruction. In the example 7 we have the first step of this procedure.

$$\text{minimize} \left\| \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 1 \end{bmatrix} - \left(\begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} h_{1n} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} h_{2n} + \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} h_{3n} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 0 \end{bmatrix} h_{4n} \right) \right| \tag{7}$$

We begin with the trivial case in which none of the basis are included in the solution. The objective function is the norm of the binary vector, which is simply the count of position equal to one. Then we calculate the gain δ of adding any column to the solution.

This gain is calculated by summing which positions in the resulting subtraction will be zero, which are the positions where both the target vector and the basis vector are one. These positions are marked with blue in equation 7. Then we subtract the superfluous positions where the basis vectors have ones and the target vector have not. Subtracting a column with a superfluous one in the target vector will increase its modulus. Those positions are marked with light red in equation 7.

$$OF = 5; \qquad h_n = [0; 0; 0; 0]; \tag{8a}$$

$$\Delta_{1n} = 2 - 1 = 1; \tag{8b}$$

$$\Delta_{2n} = 4 - 1 = 3; \tag{8c}$$

$$\Delta_{3n} = 1 - 2 = -1; \tag{8d}$$

$$\Delta_{4n} = 2 - 1 = 1 \tag{8e}$$

Then we calculate for each variable h_{gn} in $\mathcal{G}$ a gain indicator Δ_{gn}. This gain is calculated by subtracting the number of positions in which the target vector a_n has in common with the basis by the superfluous. For the next step we choose to add h_{2n} to the solution. In equation 9 we will recalculate the target vector by subtracting the added basis and repeat the procedure.

$$\text{minimize} \left\| \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ -1 \\ 0 \\ 0 \end{bmatrix} - \left(\begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} h_{1n} + \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} h_{3n} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 0 \end{bmatrix} h_{4n} \right) \right| \tag{9}$$

In this step we observe that the target vector now carries an entry equal to minus one. Even if a new basis has a one in this position, the $\ell 1$-norm of the target vector will increase because the modulus in this position will also increase. Therefore, the only case in which there is an actual gain is to cover the remaining positions in which the target vector is equal to one.

$$OF = 2; \qquad h_n = [0; 1; 0; 0]; \tag{10a}$$
$$\Delta_{1n} = 1 - 2 = -1 \tag{10b}$$
$$\Delta_{3n} = 0 - 3 = -3 \tag{10c}$$
$$\Delta_{4n} = 0 - 3 = -3 \tag{10d}$$

3.1

Also notice that, by adding a basis to the solution, the amount of remaining positions left with value one can only decrease monotonically. When we add a column to a solution tentatively, the number of ones not covered only reduces. Thus, the Δ_{gn} of the remaining columns can only monotonically decrease. This monotonicity has some interesting consequences.

Firstly, when we find a case in which all remaining Δ are negative, it is a local optimum, because Δ which are negative will never become positive later by progressively adding basis to the solution set.

Additionally, we didn't even need to consider $h_3 v$ with a respective negative Δ_{3n} from equations 8 for the second round. Since Δ_{3n} in step one depicted in equations 7 was negative, and it can only decrease, consequently, it does not need to be considered in further steps.

Also, summing positive deltas is an upper bound for any possible combination of adding g to the solution set. For instance the sum of Δ_{2n} and Δ_{4n} in the first step is two, as shown in equations 8. Consequently, when a recursive algorithm found a local optimum with objective 2 to the first state with objective 5, we know that the best we can achieve by exploring indices 2 and 4 is a objective of value 3. Thus we can stop searching and prove that the solution found in step two, as shown by equations , are optimal.

3.2 The set representation

For dealing with really large matrices, we can take advantage of the structure of the problem transforming the many binary vectors into sets that contain the positions in which this vectors are equal to one.

Let $\mathcal{I}$ a function that transforms vectors to a set of positions equal to one. We can apply this function to any rows or columns of matrices A, W, or H during the discrete basis subproblem solve.

Definition 1 (Function $\mathcal{I}$ that translates sparse vectors to sets)**.** *Let $x \in \{0,1\}^n$. Then the function $\mathcal{I}(x) : \{0,1\}^n \to 2^n$ is the set of indices of the positions of x which are greater or equal to* 1*.*

Examples of usage of function $\mathcal{I}$. Let $x = [0, 1, 1]$. Then $\mathcal{I}(x) = \{2, 3\}$. Or let $y = [1, 1, 0]$. Then $\mathcal{I}(y) = \{1, 2\}$.

An algorithm for the discrete basis subproblem using a fixed W, has to decide which positions of vector h_n should be explored fixating to 1. The algorithm starts with the trivial solution such that all positions assigned to zero. The vector h_n has size $|\mathcal{G}|$. Therefore, using the set representation, we begin the algorithm with: $\mathcal{I}(h_n) = \varnothing$ since all positions initially are zero.

We add $g \in \mathcal{G}$ to $\mathcal{I}(h_n)$ whenever we are investigating assigning position g in vector h_n to 1. As the algorithm takes steps t it inserts into the current solution some basis g from $\mathcal{G}$

$$\mathcal{I}(h_n) \subset \mathcal{G}$$

Definition 2 (Set $\mathcal{Q}$ of remaining positions to cover in the target vector)**.** *The algorithm keeps track of the set $\mathcal{Q}$ of uncovered positions of the target vector a_n. The set $\mathcal{Q}$ is a subset of $\mathcal{M}$.*

$$\mathcal{Q} = \mathcal{I}(a_n) - \left(\bigcup_{g}^{\mathcal{I}(h_n)} \mathcal{I}(w_g) \right)$$

If a new g is added to solution set $\mathcal{I}(h_n)$ then we can update $\mathcal{Q}$ in the following manner:

$$\mathcal{Q} := \mathcal{Q} - \mathcal{I}(w_g)$$

Lemma 1 (Gain calculation using set representation)**.** *When solving for a_n, the increment Δ_{gn} of adding g to a solution $\mathcal{I}(h_n)$ is calculated as:*

$$\Delta_{gn} = |\mathcal{I}(w_g) \cap \mathcal{Q}| - |\mathcal{I}(w_g) - \mathcal{Q}|$$

Proof. The set $\mathcal{Q} \subset \mathcal{M}$ of uncovered positions in a_n partitions the set $\mathcal{I}(w_g)$ into two. Firstly, the intersection between $\mathcal{I}(w_g)$ and $\mathcal{Q}$ represent the uncovered positions of a_n which could be covered by adding g to the solution $\mathcal{I}(h_n)$. This would contribute positively minimizing the objective function.

The remaining elements of $\mathcal{I}(w_g)$ which are not in $\mathcal{Q}$ would contribute negatively, because they are either superfluous (do not cover any positions in a_n) or they were already covered by another group w_g during the construction of a solution.

We will give a direct proof. Suppose we add g' to the solution set $\mathcal{I}(h_n)$. Let g' as a fixed position in h_n we wish to flip to one and let $\Delta_{g'n}$ the difference in objective when changing the value of $h_{g'n}$ from zero to one. The objective function for solving just the column a_n is:

$$\text{minimize} \sum_{m \in \mathcal{M}} |a_{mn} - (\sum_{g \in \mathcal{G}} w_{mg} \cdot h_{gn})|$$

First, we remember that all elements a_{mn}, w_{mg}, h_{gn} are either $\{0, 1\}$ by definition of the problem. So, for each g the multiplication $w_{mg} \cdot h_{gn} \in \{0, 1\}$ also. Additionally, $0 \leq \sum_{g \in \mathcal{G}} w_{mg} \cdot h_{gn} \leq |\mathcal{G}|$.

The summation $\sum_{g \in \mathcal{G}} w_{mg} \cdot h_{gn}$ also can be simplified to $\sum_{g \in \mathcal{I}(h_n)} w_{mg} \cdot h_{gn}$ for a given solution $\mathcal{I}(h_n)$, because if $g \notin \mathcal{I}(h_n)$ then $h_{gn} = 0$.

The term $a_{mn} - (\sum_{g \in \mathcal{G}} w_{mg} \cdot h_{gn})$ can assume values from 1 to $-|\mathcal{G}|$.

$a_{mn} - (\sum_{g \in \mathcal{G}} w_{mg} \cdot h_{gn})$	1,	0,	-1,	-2,	...,	$-\|\mathcal{G}\|$
$\|a_{mn} - (\sum_{g \in \mathcal{G}} w_{mg} \cdot h_{gn})\|$	1,	0,	1,	2,	...,	$\|\mathcal{G}\|$

So, when $h_{g'n}$ becomes one, for all t in which $w_{mg'}$ is one will have the summation $(\sum_{g \in \mathcal{G}} w_{mg} \cdot h_{gn})$ increase also one in value. Which means go right on the above scale. So, the only possibility for minimizing the objective function is to use the summation to cover a position where $a_{mn} = 1$. Then the module will decrease from 1 to 0. Otherwise the module will only increase.

For each position $t \in \mathcal{I}(w_{g'})$, the change in the value of the term $|a_{mn} - (\sum_{g \in \mathcal{G}} w_{mg} \cdot h_{gn})|$ will fall into 3 cases:

Case 1: $a_{m'n} = 0$
only the summation over $\mathcal{G}$ increases, so the objective function also increases.

Case 2: $a_{m'n} = 1$ and $(\sum_{g \in \mathcal{I}(h_n)} w_{mg} \cdot h_{gn}) > 0$ in this case, the summation has enough magnitude to cancel out $a_{mn} = 1$, so the increase in the summation increases the objective function

Case 3: $a_{m'n} = 1$ and $(\sum_{g \in \mathcal{I}(h_n)} w_{mg} \cdot h_{gn}) = 0$. Only in this case, when the summation is equal to 0 in that position that the absolute value decreases, because it will *cover* the position a_{mn}. Therefore, the only thing that we must track is the uncovered positions a_{mn}. We will do it maintaining a set $\mathcal{Q} \subset \mathcal{I}(a_n) \subset \mathcal{M}$ in which we deduce the positions t in which $a_{mn} = 1$ and the summation over $\mathcal{G}$ is still equal to 0, this means, is not covered. □

With this lemma, we can now prove the main foundation for this work. The following theorem will allow for an efficient search in practice.

Theorem 2 (Contribution decreasing monotonicity)**.** *Whenever adding g to solution $\mathcal{I}(h_n)$, all the gains of adding any other element in the solution in next steps can only stay the same or decrease. Which means when recalculating all other $\Delta_{g'n}$ of g' not yet in the solution set $\mathcal{I}(h_n)$, the new value is lesser or equal than it was before.*

Proof. By Lemma 1, we have that $\Delta_{gn} = |\mathcal{I}(w_g) \cap \mathcal{Q}| - |\mathcal{I}(w_g) - \mathcal{Q}|$.
When you add $\{g\}$ to solution set $\mathcal{I}(h_n)$, we update the set $\mathcal{Q}$ by subtracting the newly covered positions. So, for the remaining positions g', the $\Delta_{g'n}$ is updating taking into consideration that $\mathcal{Q} := \mathcal{Q} - \mathcal{I}(w_g)$. Therefore, $\mathcal{Q}$ has fewer items than before, and $|\mathcal{I}(w_g) \cap \mathcal{Q}|$ becomes less or equal than before and $|\mathcal{I}(w_g) - \mathcal{Q}|$ becomes greater or equal than before. Consequently, the value of Δ_{gn} can only decrease or stay the same. □

Theorem 2 has many interesting consequences. The monotonicity can be used to define local optima and to eliminate positions to search during a recursive enumeration. This enables the design of a backtracking algorithm which finds the global optimum for the sub-problem and only explores a small subset of the combinatorial decision space.

Corollary 3 (Local optimum and Negative contributing candidates skipping)**.** *If $\Delta_{g'n}$ associated with any remaining $g' \notin \mathcal{I}(h_n)$ is negative, then g' will never be in any local optimum solution with the g that belong to the current solution set $\mathcal{I}(h_n)$.*

Proof. Since the $\Delta_{g''n}$ of every position g'' only decreases when adding any other g' to the solutions set $\mathcal{I}(h_n)$, then adding g' with negative $\Delta_{g'n}$ would not only leave the objective function worse, but it would also worsen all the other $\Delta_{g''n}$, the potential of constructing better solutions.

So, there always exists a solution better than one constructed by adding g', a solution that simply skipped adding g' will stay ahead. Therefore, a solution containing g' could not be a local optimum, nor a global optimum consequently. □

Corollary 4 (Early stopping upper-bound)**.** *The solution set $\mathcal{I}(h_n)$ is a subset of $\mathcal{G}$. Any subset $\mathcal{P}$ of $\mathcal{G}$ disjoint from $\mathcal{I}(h_n)$ can have it's overall upper-bound calculated as.*

$$\Delta_{UB} = \sum_{g \in \mathcal{P}} max(\Delta_{gn}, 0)$$

Which means if any subset of $\mathcal{P}$ is added to the solution set $\mathcal{I}(h_n)$, the overall contribution to the objective function is bounded by Δ_{UB}.

Proof. The *max* function is just a mechanism to select the positive Δ_{gn}. So suppose that any $g \in \mathcal{P}$ is added to solution set $\mathcal{I}(h_n)$ then by theorem 1, all the remaining Δ_{gn} are updated to be of a lesser or equal value. Then, the sum of all the positive $\Delta_g v$ for all g before adding is greater than the actual Δ_{gn} at the point of adding them to the solution, and updating the objective function.

Thus, Δ_{UB} is greater than the overall gain of adding of any subset of $\mathcal{P}$ in any order. Consequently, if $\mathcal{P}$ is the set of remaining candidate g to explore, and we know that exists a solution OF^* lesser than the current one $OF - \Delta_{UB}$, we don't need to explore $\mathcal{P}$. Because any solution would be worse than the one with value OF^*. □

If we assume a sequential inclusion of candidate bases to the solution, at any given point during the search, we can sum positive deltas remaining to explore and calculate an upper-bound of the contribution of any combination of insertions of the associated bases. This means that if we already know any solution, this fact can be used to prove that we don't need to further explore a significant part of the decision space.

3.3 The example revisited

With the theoretical basis we can revisit the first example using the set representation. Given the matrix A, we obtain using any means an approximation for W, which we treat as fixed for the discrete basis subproblem. For a target column a_n, we search for the best combinations of the bases w_1 through w_4 that, when summed, are the best approximation for it. We apply the function $\mathcal{I}$ to each of the columns of W and target vector a_n, which is a column of A.

$$\mathcal{I}(a_n) = \{1, 5, 7, 9, 10\} \tag{11a}$$
$$\mathcal{I}(w_1) = \{1, 3, 5\} \tag{11b}$$
$$\mathcal{I}(w_2) = \{5, 7, 8, 9, 10\} \tag{11c}$$
$$\mathcal{I}(w_3) = \{2, 4, 7\} \tag{11d}$$
$$\mathcal{I}(w_4) = \{7, 8, 9\} \tag{11e}$$
$$\tag{11f}$$

Then we calculate the gain of adding each of the bases to the solution using Lemma 1.

$$\mathcal{I}(h_n) := \varnothing \tag{12a}$$
$$\mathcal{Q} := \mathcal{I}(a_n) = \{1, 5, 7, 9, 10\} \tag{12b}$$
$$OF := \|\mathcal{I}(a_n)\| = 5 \tag{12c}$$
$$\Delta_{1n} = \|\mathcal{I}(w_1) \cap \mathcal{Q}\| - \|\mathcal{I}(w_1) - \mathcal{Q}\| = \|\{1, 5\}\| - \|\{3\}\| = 1 \tag{12d}$$
$$\Delta_{2n} = \|\mathcal{I}(w_2) \cap \mathcal{Q}\| - \|\mathcal{I}(w_2) - \mathcal{Q}\| = \|\{5, 7, 9, 10\}\| - \|\{8\}\| = 3 \tag{12e}$$
$$\Delta_{3n} = \|\mathcal{I}(w_3) \cap \mathcal{Q}\| - \|\mathcal{I}(w_3) - \mathcal{Q}\| = \|\{7\}\| - \|\{2, 4\}\| = -1 \tag{12f}$$
$$\Delta_{4n} = \|\mathcal{I}(w_2) \cap \mathcal{Q}\| - \|\mathcal{I}(w_2) - \mathcal{Q}\| = \|\{7, 9\}\| - \|\{8\}\| = 1 \tag{12g}$$

Observe in equations 12 that we already have position 3 with a negative delta. Since the contributions are monotonically decreasing, we do not need to consider it again in further steps.

$$\text{Second call} \tag{13a}$$
$$\mathcal{I}(h_n) := \mathcal{I}(h_n) \cup \{2\} = \{2\} \tag{13b}$$
$$\mathcal{Q} := \mathcal{Q} - \mathcal{I}(w_2) = \{1\} \tag{13c}$$
$$OF := OF - \Delta_{2n} = 5 - 3 = 2 \tag{13d}$$
$$\Delta_{1n} := \|\mathcal{I}(w_1) \cap \mathcal{Q}\| - \|\mathcal{I}(w_1) - \mathcal{Q}\| = \|\{1\}\| - \|\{3, 5\}\| = -1 \tag{13e}$$
$$\Delta_{4n} := \|\mathcal{I}(w_2) \cap \mathcal{Q}\| - \|\mathcal{I}(w_2) - \mathcal{Q}\| = \|\varnothing\| - \|\{7, 8, 9\}\| = -3 \tag{13f}$$

In equations 13, we added base 1 to solution and eliminated base 3 because it had a negative contribution. The remaining bases, 1 and 4, now have negative contribution. Therefore, there is no way of further adding any basis to the solution without worsening the objective function. In a recursive scheme, the procedure should go back to the state described by equations 12. Now the differences are that we already explored adding 2 to the solution set $\mathcal{I}(h_n)$ and that there is a local optimum with objective function equals to 2.

The natural approach is to choose between bases 1 and 4 to begin a new search. However, both their respective Δ's, when summed up, are equal to 2, which we know is an upper-bound for the contribution of adding them in any combination in any order. Since the current objective functions is back to 5, and the contribution upper-bound is 2, the best we can expect by adding these bases is 3, which is more than the solution we already found with just the basis 2. Therefore, we also do not need to search using bases 1 and 4. Finally, we can conclude that the local optimum we found was actually the global optimum.

4 Discussion

We presented the theoretical background for reducing the space of the exhaustive search for solutions in the discrete basis problem. A recursive algorithm could explore the solution space by adding positions to the solution set progressively, initiating in the trivial solution.

By the theorem 2 and the corollary 3, it can only consider positive deltas reducing drastically the height of the recursion tree. Not only it can consider positive deltas in the first recursion, but in every recursion. Since the contributions are monotonically decreasing, each time the algorithms adds a new basis to the solution set, all other remaining Δ's tend to decrease. Therefore, each step, the number of candidates to consider diminishes.

Also, the corollary 4 can eliminate in every recursion the need to explore a combination of inclusions, given that the sum of current Δ's is an upperbound for the inclusion of any combination of the remaining candidates. Therefore we expect to have a gain in the branching factor of an exhaustive search.

References

[1] Kovacs, R. A., Gunluk, O., and Hauser, R. A. Binary matrix factorisation via column generation. In *Proceedings of the AAAI Conference on Artificial Intelligence* (2021), vol. 35, pp. 3823–3831.

[2] Miettinen, P., Mielikäinen, T., Gionis, A., Das, G., and Mannila, H. The discrete basis problem. *IEEE transactions on knowledge and data engineering 20*, 10 (2008), 1348–1362.

[3] Mirisaee, H., Gaussier, E., and Termier, A. Efficient local search for $l_1 and l_2 binary matrix factorization. Intelligent Data Analysis (2016), 783 - -807$.

[4] Snael, V., Platoa¡, J., and Kromer, P. Developing genetic algorithms for boolean matrix factorization. In *CEUR Workshop Proceedings* (2008), CEUR-WS, pp. 61–70.

[5] Snael, V., Platoa¡, J., Kromer, P., Hasek, D., and Frolov, A. On the road to genetic boolean matrix factorization. *Neural Network World* (2007), 675–688.