J Glob Optim (2016) 65:369–400
DOI 10.1007/s10898-015-0350-z

Heuristics for exact nonnegative matrix factorization

Arnaud Vandaele[1] · Nicolas Gillis[1] · François Glineur[2,3] · Daniel Tuyttens[1]

Received: 26 November 2014 / Accepted: 8 August 2015 / Published online: 5 September 2015

Abstract The exact nonnegative matrix factorization (exact NMF) problem is the following: given an m-by-n nonnegative matrix X and a factorization rank r, find, if possible, an m-by-r nonnegative matrix W and an r-by-n nonnegative matrix H such that $X = WH$. In this paper, we propose two heuristics for exact NMF, one inspired from simulated annealing and the other from the greedy randomized adaptive search procedure. We show empirically that these two heuristics are able to compute exact nonnegative factorizations for several classes of nonnegative matrices (namely, linear Euclidean distance matrices, slack matrices, unique-disjointness matrices, and randomly generated matrices) and as such demonstrate their superiority over standard multi-start strategies. We also consider a hybridization between these two heuristics that allows us to combine the advantages of both methods. Finally, we discuss the use of these heuristics to gain insight on the behavior of the nonnegative rank, i.e., the minimum factorization rank such that an exact NMF exists. In particular, we disprove a

This paper presents research results of the Belgian Network DYSCO (Dynamical Systems, Control, and Optimization), funded by the Interuniversity Attraction Poles Programme initiated by the Belgian Science Policy Office, and of the Concerted Research Action (ARC) programme supported by the Federation Wallonia-Brussels (contract ARC 14/19-060).

✉ Arnaud Vandaele
arnaud.vandaele@umons.ac.be

Nicolas Gillis
nicolas.gillis@umons.ac.be

François Glineur
francois.glineur@uclouvain.be

Daniel Tuyttens
daniel.tuyttens@umons.ac.be

[1] Department of Mathematics and Operational Research, Faculté Polytechnique, Université de Mons, Rue de Houdain 9, 7000 Mons, Belgium

[2] Center for Operations Research and Econometrics, Université catholique de Louvain, Voie du Roman Pays, 34, 1348 Louvain-La-Neuve, Belgium

[3] ICTEAM Institute, Université catholique de Louvain, 1348 Louvain-La-Neuve, Belgium

conjecture on the nonnegative rank of a Kronecker product, propose a new upper bound on the extension complexity of generic n-gons and conjecture the exact value of (i) the extension complexity of regular n-gons and (ii) the nonnegative rank of a submatrix of the slack matrix of the correlation polytope.

Keywords Nonnegative matrix factorization · Exact nonnegative matrix factorization · Heuristics · Simulated annealing · GRASP · Hybridization · Nonnegative rank · Linear Euclidean distance matrices · Slack matrices · Extension complexity

1 Introduction

Nonnegative matrix factorization (NMF) is the problem of finding good approximations of a given nonnegative matrix as a low-rank product of two nonnegative matrices. This linear dimensionality reduction technique has been used very successfully for a large variety of machine learning and data mining tasks, including text mining and image processing [39]. Formally, given a nonnegative matrix $X \in \mathbb{R}^{m \times n}_{+}$ and a factorization rank r, NMF looks for two nonnegative matrices $W \in \mathbb{R}^{m \times r}_{+}$ and $H \in \mathbb{R}^{r \times n}_{+}$ such that $X \approx WH$. Despite the fact that NMF is NP-hard in general [53], it has been used successfully in many practical situations. A large number of dedicated nonlinear local optimization schemes have been developed to compute good factorizations [9], e.g., to try identifying good local minima of the following nonconvex optimization problem

$$\min_{W \in \mathbb{R}^{m \times r}, H \in \mathbb{R}^{r \times n}} ||X - WH||_F^2 \quad \text{such that } W \geq 0 \text{ and } H \geq 0,$$

where $||.||_F$ is the Frobenius norm. Nearly all NMF algorithms are iterative: at each step, they aim to improve the current solution. In practice, these algorithms are usually initialized randomly, or with some ad hoc strategies; see, e.g., [9,20] and the references therein.

Comparatively, much less attention has been given in the literature to the development of heuristic algorithms aimed at finding better local minima of the NMF approximation problem. In this paper, we tackle the problem of computing high quality local minima for the NMF problem. In particular, our focus is on finding exact nonnegative factorizations, that is, computing nonnegative factors W and H such that $X = WH$ holds exactly, a problem we will refer to as *exact NMF*. The minimum factorization rank for which such an exact NMF exists is called the *nonnegative rank* of X and is denoted $\text{rank}_+(X)$ [12].

1.1 Motivating applications

For machine learning and data mining applications, it typically does not make sense to look for exact NMF's because the data is usually contaminated with noise. However, several other applications are closely related to exact NMF and the nonnegative rank, including the following.

– *Computing the minimum biclique cover number of a bipartite graph* Let $G = (V = V_1 \cup V_2, E)$ be a bipartite graph with $V_1 = \{s_1, s_2, \dots, s_m\}$, $V_2 = \{t_1, t_2, \dots, t_n\}$ and $E \subseteq V_1 \times V_2$. A complete bipartite subgraph of G, referred to as a biclique, is a subgraph $G' = (V_1' \cup V_2', E')$ with $V_1' \subseteq V_1$, $V_2' \subseteq V_2$ and $E' \subseteq E$ such that $E' = V_1' \times V_2'$, that is, all vertices in V_1' and V_2' are connected. The minimum biclique cover number $\text{bc}(G)$ of G is the minimum number of bicliques needed to cover all edges in G. Let $X_G \in \{0, 1\}^{m \times n}$ be the biadjacency matrix of the graph, that is, $X_G(i, j) = 1$ for all pairs (i, j) such that $(s_i, t_j) \in E$. A biclique of G corresponds to a nonzero combinatorial rectangle in the biadjacency matrix X_G (that is, a submatrix containing only positive entries). Given an exact NMF of the biadjacency matrix $X_G = WH = \sum_k W(:,k)H(k,:)$ where $W(:,k)$ denotes the kth column of W and $H(k,:)$ the kth row of H, the nonzero pattern of each rank-one factor $W(:,k)H(k,:)$ must correspond to a biclique. In fact, because W and H are nonnegative, $X_G(i,j) = 0 \Rightarrow W(i,k)H(k,j) = 0$ for all k where $X_G(i,j)$ denotes the entry of X_G at position (i,j). Moreover, the union of the bicliques corresponding to the rank-one factors must cover G since $X_G = WH$. Therefore,

$$\mathrm{bc}(G) \leq \mathrm{rank}_+(X_G),$$

and computing an exact NMF of X_G provides an upper bound for the minimum biclique cover of G (although this bound for $\mathrm{bc}(G)$ based on the nonnegative rank does not seem to have been used much in the literature). Conversely, a minimum biclique cover of G provides a lower bound for the nonnegative rank of X_G, which is referred to as the *rectangle covering bound* and is denoted $\mathrm{rc}(X_G) = \mathrm{bc}(G) \leq \mathrm{rank}_+(X_G)$; see [16] and the references therein.

– *Computing the extension complexity of polyhedra* Given a polytope $\mathcal{P}$, an extension (or lift, or extended formulation) of $\mathcal{P}$ is a higher-dimensional polytope $\mathcal{Q}$ for which there exists a linear projection π such that $\pi(\mathcal{Q}) = \mathcal{P}$. If the number of facets of $\mathcal{P}$ is large (possibly growing exponentially with the dimension k), a crucial question in combinatorial optimization is whether there exists an extension with a small number of facets (ideally bounded by a polynomial in the dimension k), in which case a linear program over $\mathcal{P}$ can potentially be solved much more effectively using an equivalent formulation over $\mathcal{Q}$. The minimum number of facets appearing in any extension of $\mathcal{P}$ is called the extension complexity of $\mathcal{P}$. In [55], Yannakakis proved that the extension complexity of a polytope $\mathcal{P}$ is equal to the nonnegative rank of its slack matrix $S_{\mathcal{P}}$ (see Sect. 2 for a definition of the slack matrix of a polytope). It is also worth mentioning that any exact NMF of $S_{\mathcal{P}}$ provides an explicit extension for $\mathcal{P}$. This result has been extensively used recently to prove bounds on the extension complexity of well-known polytopes; see [13,28,35] and the references therein. For example, it was shown very recently that the perfect matching polytope admits no polynomial-size extension [46], answering a long-standing open question in combinatorial optimization.
– *Conjecturing new theoretical results on the nonnegative rank, or disproving them* As any exact NMF of a nonnegative matrix provides an upper bound on its nonnegative rank, one can use this technique to find counter-examples to conjectures dealing with the nonnegative rank of matrices, or strengthen our belief that some conjectures are correct. For example, Beasley and Laffey [2] developed some lower bounding techniques for the nonnegative rank of n-by-n linear Euclidean distance matrices (see Sect. 3 for more details) and conjectured that these rank-three matrices have nonnegative rank n. Using a standard NMF algorithm combined with a simple multi-start heuristic, Gillis and Glineur [23] found a counterexample: the 6-by-6 linear Euclidean distance matrix $M(i,j) = (i-j)^2$ $(1 \leq i, j \leq 6)$ has nonnegative rank five, which disproved the conjecture and motivated the development of stronger lower bounds for the nonnegative rank of such matrices. Along the same line, Hrubeš [31] developed some new upper bounding techniques for the nonnegative rank of such matrices. In Sect. 6, we discuss several examples where the use of exact NMF algorithms allows us to gain insight on the nonnegative rank.

Other problems closely related to nonnegative rank and exact NMF computations arise in communication complexity [41], probability [8] and computational geometry [23]; see also, e.g., [20] and the references therein.

1.2 Computational complexity

Given an m-by-n nonnegative matrix X, Vavasis [53] proved that checking whether $\text{rank}(X) = \text{rank}_+(X)$ is NP-hard. Therefore, unless $P = NP$, no algorithm can decide whether $\text{rank}(X) = \text{rank}_+(X)$ using a number of arithmetic operations bounded by a polynomial in m, n and $\text{rank}(X)$. Nevertheless, Arora et al. [1] showed that checking whether a nonnegative matrix admits an exact rank-r NMF can be done in time polynomial in m and n (i.e., considering the factorization rank r fixed). This result relies on a clever reformulation of the exact NMF problem for an $m \times n$ matrix as a system of $\mathcal{O}(mn)$ fixed-degree polynomial equalities involving $\mathcal{O}(r^2 2^r)$ variables. This, combined with the fact that a system of k polynomial inequalities up to degree d and in p variables can be solved in $\mathcal{O}((kd)^p)$ operations, shows that checking the existence of an exact rank-r NMF can be done with total complexity $\mathcal{O}((mn)^{r^2 2^r})$.

This complexity was later improved by Moitra [42] to $\mathcal{O}((mn)^{r^2})$. Unfortunately, because they rely on quantifier elimination, these results do not translate into practical algorithms, even when dealing with very small matrices. For example, we were unable to compute a rank-three NMF of a 4-by-4 matrix (which is actually the first non-trivial case since $\text{rank}(X) = 2 \Leftrightarrow \text{rank}_+(X) = 2$ [12,51]) using either the built-in polynomial equation solver of Mathematica (which runs out of memory after performing a large number of operations) or the qepcad software [7] dedicated to quantifier elimination.

It is therefore not clear whether these theory-oriented complexity results can prove useful to perform exact NMF, even for small-scale matrices, which prompted us to introduce the use of heuristics to tackle the problem.

Remark 1 (Separability) Note that Arora et al. [1] also discuss the separabilty condition that makes the NMF problem easily solvable. This condition states that there exists an exact factorization where all columns of the first factor also appear as columns of the input matrix. Although this condition makes sense in several applications (see, e.g., [24] and the references therein), it is rather strong and is in general not satisfied. In particular, it is not satisfied by the matrices considered in this paper.

1.3 Contribution and outline of the paper

The paper is organized as follows. Section 2 lists the classes of nonnegative matrices on which we benchmark exact NMF algorithms, and provides a description of the corresponding applications. Section 3 presents two multi-start strategies, and compares their combination with several initialization strategies and state-of-the-art NMF algorithms. This allows us to select an NMF algorithm (that is, a method to locally improve a current solution) and initialization strategies for the rest of the paper. Section 4 introduces two heuristics dedicated to exact NMF (SA and RBR) along with a hybridization. Section 5 compares these heuristics, showing that they outperform multi-start strategies. In particular, RBR performs remarkably well and is able to identify exact NMF's very efficiently for several classes of matrices, while SA and the hybridization strategy are able to compute an exact NMF for all considered matrices. Section 5 also discusses the limitations of these approaches, which are unable to compute exact NMF's for large and difficult matrices (as expected by the computational complexity of the problem). Finally, Sect. 6 discusses the use of these heuristics to better understand the nonnegative rank. In particular, we propose new conjectures for the nonnegative rank of (i) the Kronecker product of two nonnegative matrices, (ii) the slack matrices of regular and generic n-gons, and (iii) a submatrix of the slack matrix of the correlation polytope.

To summarize, the main contributions of the paper are threefold:

– Design of two heuristics for exact NMF, along with a hybridization strategy, that outperform multi-start strategies. To the best of our knowledge, the only heuristic algorithms previously designed for NMF were developed in [32–34] and focused only on the (approximate) NMF problem, and not its exact counterpart.
– Comparison of these heuristics with two simple multi-start strategies on several classes of nonnegative matrices for which exact factorizations are relevant for applications. This is to the best of our knowledge the first time exact NMF algorithms are benchmarked on this type of nonnegative matrices (previous work focused on randomly generated matrices, or on machine learning data sets for which exactness of the factorization is not relevant).
– Several examples of the concrete use of these heuristics to address open theoretical questions related to the nonnegative rank.

The code and data sets used in the paper have been made available online at

https://sites.google.com/site/exactnmf

We hope that the promising results shown by the methods introduced in this paper will motivate researchers to further develop even faster and more effective heuristics for exact NMF.

2 Benchmark nonnegative matrices for exact NMF

Throughout the paper, we will compare exact NMF algorithms on the following nonnegative matrices (see Table 1):

– *Linear Euclidean distance matrices* Given a set of real numbers a_i for $1 \leq i \leq n$, a linear Euclidean distance matrix (EDM) is defined as

$$X_a(i, j) = (a_i - a_j)^2 \quad \text{for } 1 \leq i \leq n \text{ and } 1 \leq j \leq n.$$

If at least three entries of $a \in \mathbb{R}^n$ are distinct, then $\text{rank}(X_a) = 3$. However, the nonnegative rank of linear EDM's can be arbitrarily large: in fact, it was shown in [2] using results from [14,30] that, if the entries of a are distinct,

$$\text{rank}_+(X_a) \quad \geq \quad \min \left\{ k \;\middle|\; \binom{k}{\lfloor k/2 \rfloor} \geq n \right\} \quad \geq \quad \log_2(n).$$

The lower bound was later improved in [23]. A subclass of linear EDM's are the following

$$X_{[n]}(i, j) = (i - j)^2 \quad \text{for } 1 \leq i \leq n \text{ and } 1 \leq j \leq n,$$

for which it was proved in [31, Th.1] that

$$\text{rank}_+(X_{[2n]}) \leq \text{rank}_+(X_{[n]}) + 2.$$

If n is a power of two, we therefore have $\text{rank}_+(X_{[n]}) \leq 2\log_2(n)$ since $\text{rank}_+(X_{[2]}) = 2$. Combining this upper bound with the lower bound from [23] allows us to determine the nonnegative rank for these matrices up to $n = 16$; see Table 1. However, as we will see later on, it is non-trivial to compute exact NMF for these matrices.

Table 1 Nonnegative matrices used to compare exact NMF heuristics

	m	n	Rank(X)	Rank$_+(X)$	Abbreviation
Linear EDM's $X(i, j) = (i - j)^2$, for $1 \le i \le m,\ 1 \le j \le n$	6	6	3	5	LEDM6
	8	8	3	6	LEDM8
	12	12	3	7	LEDM12
	16	16	3	8	LEDM16
	32	32	3	10*	LEDM32
Slack matrix of the hexagon	6	6	3	5	6-G
Slack matrix of the heptagon	7	7	3	6	7-G
Slack matrix of the octagon	8	8	3	6	8-G
Slack matrix of the nonagon	9	9	3	7	9-G
Slack matrix of the hexadecagon	16	16	3	8	12-G
Slack matrix of the 32-gon	32	32	3	10	32-G
Slack matrix of the dodecahedron	20	12	4	9	20-D
Slack matrix of the 24-cell	24	24	5	12*	24-C
UDISJ ($n = 4$)	16	16	9	9	UDISJ4
UDISJ ($n = 5$)	32	32	18	18	UDISJ5
UDISJ ($n = 6$)	64	64	27	27	UDISJ6
Randomly generated matrices: $X = WH$					
`Density = 0.1`	50	50	10	10	RND1
`Density = 0.3`	50	50	10	10	RND3

The symbol * means that the exact value of the nonnegative rank is still unknown, i.e., the best known lower bound does not match the best known upper bound (for LEDM 32, the best lower bound is 9 while for 24-C it is 10). However, after running our heuristics extensively on these matrices, we believe that all values of the nonnegative ranks appearing in this table are correct

- *Slack matrices* The slack matrix of a polytope $\mathcal{P}$ with m facets and n vertices is defined as the $m \times n$ nonnegative matrix $S_\mathcal{P}$ whose (i, j)th entry $S_\mathcal{P}(i, j)$ is equal to the slack of the jth vertex with respect to the ith facet. Formally, given the list of n vertices v_j $(1 \le j \le n)$ and a facet description of the polytope $\mathcal{P} = \{x \in \mathbb{R}^k \mid A(i, :)x \le b_i \text{ for } 1 \le i \le m\}$, we have that

$$S_\mathcal{P}(i, j) = b(i) - A(i, :)v_j \ge 0 \text{ for } 1 \le i \le m \text{ and } 1 \le j \le n.$$

  As recalled in the introduction, the nonnegative rank of the slack matrix of $\mathcal{P}$ is equal to the extension complexity of $\mathcal{P}$. In this paper, we use slack matrices of several well-known classes of polytopes; see Table 1 (note that, unless stated otherwise, each mention of an n-gon in this paper refers to the regular n-gon).
- *Unique-disjointness matrices* A unique-disjointness (UDISJ) matrix $X_n \in \{0, 1\}^{2^n \times 2^n}$ of order n is a matrix whose rows and columns are indexed by all vectors in $a, b \in \{0, 1\}^n$ and which satisfies

$$X_n(a, b) = \begin{cases} 1 & \text{if } a^T b = 0, \\ 0 & \text{if } a^T b = 1, \\ ? & \text{otherwise,} \end{cases}$$

where ? means that the corresponding entry can be 0 or 1. UDISJ matrices have been successfully used to prove lower bounds on the extension complexity of polytopes, because their sparsity pattern can be found in submatrices of several interesting slack matrices; see, e.g., [36] and the references therein. Note that UDISJ matrices also often appear in the communication complexity literature.[1] Many of the best lower bounds for UDISJ matrices are based on the rectangle covering bound (see Sect. 1), and the class of matrices we consider here is built on a similar principle (we choose not to use the UDISJ matrices themselves as their nonnegative rank is not known exactly).
Given r rank-one binary (combinatorial) rectangles $w_k h_k^T$ defined with some $w_k, h_k \in \{0,1\}^{2^n}$ $(1 \leq k \leq r)$ covering X_n, we define

$$Y_n = \sum_{k=1}^{r} w_k h_k^T \; \in \; \{0, 1, \ldots, r\}^{2^n \times 2^n}.$$

The matrix Y_n features the same sparsity pattern as X_n (that is, $Y_n(i,j) \neq 0 \Leftrightarrow X_n(i,j) \neq 0$ for all i, j). We verified numerically that $\text{rank}(Y_n) = r$ and since these matrices clearly admit a rank-r NMF, we can conclude that $\text{rank}_+(Y_n) = r$, and we will use those matrices Y_n for our benchmark; see Table 1.

– *Randomly generated matrices* It is standard in the NMF literature to use randomly generated matrices to compare algorithms (see, e.g., [38]), with the nice feature that the resulting nonnegative rank of these matrices can be specified. For example, generating each entry of $W \in \mathbb{R}^{m \times r}$ and $H \in \mathbb{R}^{r \times n}$ uniformly at random in the interval [0,1] and computing $X = WH$ generates, with probability one, a nonnegative matrix X such that $\text{rank}(X) = \text{rank}_+(X) = r$. In this paper, we have generated such matrices of dimensions 50-by-50 with nonnegative rank 10. More precisely, the matrix W is generated as follows:

  (i) generate W such that each column of the 50-by-10 matrix W has exactly one non-zero entry whose location is randomly chosen and its value is picked uniformly at random in the interval [0,1] (this ensures each rank-one factor to be non-zero), and
  (ii) add a sparse uniformly distributed (in the interval [0,1]) random update to W, with prespecified density d (that is, apply `W = W + sprand(50,10,d)`).

  We use $d = 0.1$ and 0.3 as specified in Table 1. Matrix H is generated in the same way. It turns out that these nonnegative products WH are relatively easy to factorize: in fact, most initializations lead most NMF algorithms to an exact NMF. Hence these matrices are not very useful to compare exact NMF heuristics ; nevertheless we include them in our comparisons to illustrate this fact. (Note that this procedure may generate rank-deficient matrices with a nonzero probability, in particular if two columns of W have only one non-zero element at the same position. However, we have made sure that the factors of the matrices selected for our experiments have full rank.) (Note also that we did not simply generate each entry of W and H uniformly at random in the interval [0,1] as it is usually done. The reason is that factoring the corresponding matrix WH is usually even easier, because matrices W and H are then positive, which is known to enlarge the set of exact factorizations—we refer the reader to [23] for more information on the geometric interpretation of exact NMF. In particular, the solution of exact NMF for $M = WH$ is never unique—up to permutation and scaling—for positive matrices W and H [19].)

[1] Bob is given a, Alice b, and they have to decide whether $a^T b \neq 0$ while minimizing the number of bits exchanged; see [41] for more details.

3 Designing heuristics: key ingredients and multi-start examples

Before presenting our proposed heuristics, we explore two multi-start strategies (Sect. 3.1). This allows us to discuss some key aspects for comparing and designing such heuristics. There are four main building blocks for our proposed heuristics:

- the initialization strategy (Sect. 3.2),
- the main algorithm, i.e. the heuristic constructing exact NMF's after applying the initialization strategy, which relies on a local NMF algorithm (Sect. 3.1 for the multi-start strategies, and Sect. 4 for our two proposed heuristics),
- the NMF algorithm used to improve solutions locally (Sect. 3.3),
- a final refinement step that will try to further improve the output of the main algorithm as far as possible (ideally, until an exact NMF is found); see the description Algorithm FR, which also relies on the local NMF algorithm.

This final refinement procedure will be applied to all solutions generated by the heuristics. In this paper, we use a tolerance for the relative error equal to 10^{-6}, that is, we will assume that an exact NMF (W, H) of X is found as soon as $\frac{\|X-WH\|_F}{\|X\|_F} \leq 10^{-6}$. Algorithm FR runs a local NMF algorithm as long as the relative error decreases at least by a predefined factor α after every period of Δt seconds, otherwise it stops and returns the current solution. We set the parameters to the following rather conservative values: $\Delta t = 1$ s (which is quite large for small matrices[2]) and $\alpha = 0.99$; see "Appendix A" for some additional numerical results comparing different values for Δt and α.

Algorithm FR Final Refinement$(X, W, H, \alpha, \Delta t)$

```
Input: $X \in \mathbb{R}_+^{m\times n}$, $W \in \mathbb{R}_+^{m\times r}$, $H \in \mathbb{R}_+^{r\times n}$, $0 < \alpha < 1$, $\Delta t$.
1: $i = 1$, $e_0 = +\infty$, $e_1 = \|X - WH\|_F/\|X\|_F$.
2: while $e_i < \alpha e_{i-1}$ and $e_i > 10^{-6}$ do
3:     $i \leftarrow i + 1$.
4:     $[W, H] \leftarrow$ AlgoNMF$(X, W, H, \Delta t)$. % See Sect. 3.3
5:     $e_i \leftarrow \|X - WH\|_F/\|X\|_F$.
6: end while
7: return $W \in \mathbb{R}_+^{m\times r}$, $H \in \mathbb{R}_+^{r\times n}$, relative error $e_i$.
```

Figure 1 shows how these blocks are arranged in our design of heuristics. Since it will not be practical to display results for all possible combinations of heuristics (there will be five in total), NMF algorithms (five) and initializations (five), along with the different tuning parameters, another goal of this section is to select, for the rest of the paper, reasonable values for the parameters of a good multi-start heuristic, along with an efficient local improvement algorithm and initialization strategy that performs well on most examples. Note that although we only report results for a subset of the possible combinations, our final choice of the parameters has shown in further computations to be on average the most reliable (this will also be the case for tuning the parameters of the proposed heuristics, namely simulated annealing and the rank-by-rank heuristic; see Sect. 4).

Remark 2 *(Are our exact NMF's really exact?)* At this point, it is important to insist on the fact that all the numerical experiments performed in this paper are with floating point arithmetic.

[2] For example, for a 50-by-50 matrix and $r = 10$, running standard multiplicative updates for 1 s allows to perform about 10000 iterations on a standard laptop.

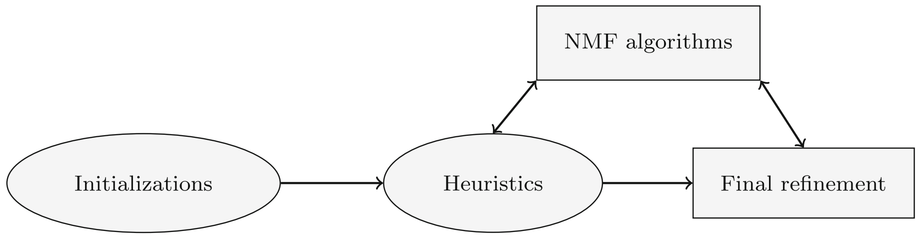

Fig. 1 Representation of the stream of the exchange of information between the different building blocks of our exact NMF heuristics. *Arrows* represent the transfer of a solution

Hence, we consider a factorization exact if the relative error $\|X - WH\|_F/\|X\|_F$ is smaller than some threshold (we choose 10^{-6}) so that the computed factorizations are not exact but high precision solutions. It is interesting to point out that all the solutions that we computed with relative error smaller than 10^{-6} could be further improved with additional iterations of Algorithm FR to 10^{-16} (which is the smallest possible using the standard Matlab precision). Note that it is an open question whether the nonnegative rank over the rationals equals the nonnegative rank over the reals; see, e.g., the discussion in [53]. Note that they were recently shown to be different for the semidefinite rank, a generalization of the nonnegative rank [27]; see [15] and the very recent [49] for more details.

3.1 Two multi-start heuristics

In this section we propose two multi-start heuristics (Sects. 3.1.1 and 3.1.2).

3.1.1 Multi-start 1

The simplest multi-start strategy one can think of is to restart Algorithm FR with many different initial matrices until an exact NMF is obtained; see Algorithm MS1. Note that this heuristic is the one used in [23] to compute exact NMF of linear EDM's. Note also that, in view of Fig. 1, MS1 corresponds to a heuristic which is an 'empty box' that transfers directly the solution from 'Initializations' to 'Final refinement'.

Algorithm MS1 Multi-Start 1$(X, r, \alpha, \Delta t)$

```
Input: $X \in \mathbb{R}^{m\times n}_+$, $r < \min(m,n)$, $0 < \alpha < 1$, $\Delta t$, tol $= 10^{-6}$.
1: $(W_0, H_0) \leftarrow$ random initialization$(m, n, r)$. % See Sect. 3.2
2: $[W, H, e] \leftarrow$ Final Refinement$(X, W_0, H_0, \alpha, \Delta t)$.
```

3.1.2 Multi-start 2

Applying Algorithm FR until convergence is useless when the error does not converge to zero, that is, when (W, H) converges to a local minimum with error strictly larger than zero. The idea behind Algorithm MS2 is to keep the pairs (W, H) with the best potential to obtaining an exact NMF, and therefore avoiding waste of computational time. The way we proceed is to generate K different random initializations, apply N iterations of an NMF algorithm to each pair and only apply Algorithm FR to the pair (W, H) with the smallest residual error among those. This heuristic can also be found in [9]. Moreover, note that MS1 is a particular case of MS2 with $K = 1$.

Algorithm MS2 Multi-Start $2(X, r, \alpha, \Delta t, K, N)$

```
Input: $X \in \mathbb{R}_+^{m\times n}$, $r < \min(m, n)$, $0 < \alpha < 1$, $\Delta t$, $K$, $N$, tol $= 10^{-6}$.

1: $e = 1$.
2: for $i = 1 \to K$ do
3:    $(\tilde{W}, \tilde{H}) \leftarrow$ random initialization$(m, n, r)$. % See Sect. 3.2
4:    $[\tilde{W}, \tilde{H}] \leftarrow$ AlgoNMF$(X, \tilde{W}, \tilde{H}, N)$. % See Sect. 3.3
5:    $\tilde{e} = \|X - \tilde{W}\tilde{H}\|_F / \|X\|_F$.
6:    if $\tilde{e} < e$ then
7:       $(W, H) \leftarrow (\tilde{W}, \tilde{H})$.
8:       $e \leftarrow \tilde{e}$.
9:    end if
10: end for
11: $[W, H] \leftarrow$ Final Refinement$(X, W, H, \alpha, \Delta t)$.
```

3.1.3 Comparing the multi-start heuristics

Table 2 gives the computational results for the two multi-start heuristics with different parameters for MS2 (namely, $N = 20, 40$ and $K = 100, 200$). Throughout the paper (unless stated otherwise), the settings are the following:

- We use the same randomly generated initial matrices to obtain a fair comparison between the different runs (and for the results to be reproducible). In order to do so, we will control the random number generator of Matlab as follows: it is initialized with the value 1 (that is, we execute `rng(1)`) and after each outer loop of the heuristics (for example, after step 2 of MS1 and MS2), it is increased by one (that is, we execute `rng(i+1)` where i is the number of iterations performed so far).
- We perform at most 100 runs of each heuristic. In order to reduce the computational time of the numerical experiments, we stop testing a given heuristic as soon as (at least) five exact NMF's for a given nonnegative matrix have been found (this condition being checked after every ten runs).
- The tables display the number of exact NMF's found out of the number of runs performed (for example, 6/10 means that the algorithm found six exact NMF's out of ten runs). They also display in brackets the average time in seconds needed to compute a single exact NMF. The best results in terms of average running time are underlined, and the best heuristics in term of robustness (i.e. proportion of exact factorizations found) are in bold; see the caption of Table 2 for more details.

We refer the reader to Sect. 3.3 for the local search NMF algorithm selected and to Sect. 3.2 for the initialization strategy of matrices W and H. All tests are preformed using Matlab on a PC Intel CORE i5-4570 CPU @3.2GHz $\times$ 4, with 7.7Go of RAM.

We observe in Table 2 that MS2 *performs better* than MS1, while the variants of MS2 perform similarly. Note that the computational times are rather similar: the reason is that the considered matrices are rather small and performing N iterations of the NMF algorithm is therefore relatively quick. In the remainder of the paper, we will use the parameters $K = 200$

Table 2 Comparison of the multi-start heuristics

	MS1	MS2(100,20)	MS2(200,20)	MS2(100,40)	MS2(200,40)
LEDM6	5/80 (35)	9/20 (4.7)	7/10 (3.2)	11/20 (4.4)	**9/10** (3.2)
LEDM8	0/100 (∼)	**6/50** (29.2)	**6/40** (20)	5/50 (35.8)	**6/40** (37)
LEDM12	0/100 (∼)	0/100 (∼)	0/100 (∼)	0/100 (∼)	0/100 (∼)
LEDM16	0/100 (∼)	0/100 (∼)	0/100 (∼)	0/100 (∼)	0/100 (∼)
LEDM32	0/100 (∼)	0/100 (∼)	0/100 (∼)	0/100 (∼)	0/100 (∼)
6-G	8/30 (6.6)	**10/10** (1.6)	**10/10** (1.8)	**10/10** (1.9)	**10/10** (2.9)
7-G	8/20 (4.1)	**9/10** (1.9)	**10/10** (2.2)	**9/10** (2.3)	**10/10** (3)
8-G	5/60 (23.3)	6/10 (3.2)	**9/10** (2.3)	**9/10** (2.3)	**10/10** (3)
9-G	7/40 (10.6)	7/10 (2.8)	6/10 (4.2)	6/10 (4.1)	**8/10** (4.2)
16-G	0/100 (∼)	0/100 (∼)	0/100 (∼)	0/100 (∼)	0/100 (∼)
32-G	0/100 (∼)	0/100 (∼)	0/100 (∼)	0/100 (∼)	0/100 (∼)
20-D	0/100 (∼)	0/100 (∼)	0/100 (∼)	0/100 (∼)	0/100 (∼)
24-C	0/100 (∼)	0/100 (∼)	0/100 (∼)	0/100 (∼)	0/100 (∼)
UDISJ4	6/10 (2.4)	**10/10** (1.8)	**10/10** (2.1)	**10/10** (2.2)	**10/10** (3.4)
UDISJ5	5/30 (12.2)	5/30 (30.1)	5/30 (36.4)	**9/20** (14.9)	**5/10** (24.2)
UDISJ6	**2/100** (119.5)	0/100 (∼)	0/100 (∼)	**1/100** (1211.9)	0/100 (∼)
RND1	**10/10** (1.1)	**10/10** (1.9)	**10/10** (2.3)	**10/10** (2.6)	**10/10** (4.1)
RND3	**10/10** (1.1)	**10/10** (2)	**10/10** (2.4)	**10/10** (2.5)	**10/10** (4)

The ratio x/y means that x exact NMF's have been found out of y runs of the heuristic, while the number in brackets is the average running time for a heuristic to find a single exact NMF. *Underlined*: (i) the best heuristic in terms of average running time to compute a single exact NMF, and (ii) any heuristic whose running time to compute an exact NMF is at most 10 % away from the best heuristic. In *bold*: (i) the best heuristic in terms of number of exact NMF's found out of a given number of runs, and (ii) any heuristic which is at most 10 % away from the best heuristic. Similar conventions are used for subsequent Tables 3–6 and 8–16

and $N = 20$ for MS2 as it offers a good compromise between proportion of exact NMF's found and total computational time.

It is interesting to note that

- For randomly generated matrices, as already anticipated in Sect. 2, all heuristics are able to identify an exact NMF for all runs.
- For some linear EDM's (LEDM12 to LEDM32) and slack matrices (16-G to 24-C), no multi-start strategy is able to identify an exact NMF. This observation is the main motivation to develop more efficient heuristics for exact NMF: we had to run MS1 for several hours (which means thousands of initializations) to find an exact rank-12 NMF of 24-C (slack matrix of the 24-cell).

3.2 Selecting an initialization strategy

In this section, we describe several random initialization strategies. The most widely used strategy is to generate each entry of the initial W and H factors uniformly at random in the interval [0,1], a strategy which we refer to as RNDCUBE. As we will see, RNDCUBE performs rather poorly, and we propose a new very effective random initialization strategy which allows to explore the search domain in a much better way. In fact, the issue with

Table 3 Comparison of the different initialization strategies combined with multi-start 2

	Sparse 00	Sparse 10	Sparse 01	Sparse 11	RNDCUBE
LEDM6	5/100 (54.8)	5/90 (50.6)	**8/10** (2.7)	7/10 (3.2)	6/60 (25.1)
LEDM8	4/100 (113.3)	3/100 (133.5)	**6/30** (23.5)	**6/40** (20)	0/100 (∼)
LEDM12	0/100 (∼)	0/100 (∼)	0/100 (∼)	0/100 (∼)	0/100 (∼)
LEDM16	0/100 (∼)	0/100 (∼)	0/100 (∼)	0/100 (∼)	0/100 (∼)
LEDM32	0/100 (∼)	0/100 (∼)	0/100 (∼)	0/100 (∼)	0/100 (∼)
6-G	**10/10** (1.7)	**10/10** (1.9)	**10/10** (2.1)	**10/10** (1.8)	**10/10** (2)
7-G	**10/10** (1.9)	7/10 (2.9)	**10/10** (2.2)	**10/10** (2.2)	**10/10** (1.9)
8-G	6/10 (4.1)	6/10 (3.8)	**9/10** (2.5)	**9/10** (2.3)	5/30 (16.7)
9-G	8/20 (6.3)	5/30 (16.1)	5/10 (4.9)	**6/10** (4.2)	5/40 (23.3)
16-G	0/100 (∼)	0/100 (∼)	0/100 (∼)	0/100 (∼)	0/100 (∼)
32-G	0/100 (∼)	0/100 (∼)	0/100 (∼)	0/100 (∼)	0/100 (∼)
20-D	0/100 (∼)	0/100 (∼)	0/100 (∼)	0/100 (∼)	0/100 (∼)
24-C	0/100 (∼)	0/100 (∼)	0/100 (∼)	0/100 (∼)	0/100 (∼)
UDISJ4	**10/10** (2.2)	**10/10** (2.1)	**10/10** (2.3)	**10/10** (2.1)	**10/10** (2.1)
UDISJ5	**7/20** (17.1)	6/20 (20.4)	6/30 (30.5)	5/30 (36.4)	**7/20** (16.5)
UDISJ6	**2/100** (465.9)	**5/80** (146.3)	**1/100** (921.8)	0/100 (∼)	0/100 (∼)
RND1	**10/10** (2.3)	**10/10** (2.4)	**10/10** (2.8)	**10/10** (2.3)	**10/10** (2.1)
RND3	**10/10** (2.5)	**10/10** (2.3)	**10/10** (2.2)	**10/10** (2.4)	**10/10** (2.3)

generating each entry of W and H uniformly at random in the interval [0,1] is that it only generates dense matrices, while it is well-known that

(i) exact NMF solutions usually have many zero entries (see, e.g., the discussion in [21]), and
(ii) the boundary of the feasible domain only contains sparse matrices ; hence generating only dense initial matrices starts the exploration relatively far away from that boundary where solutions are in general located.

The sparsest possible way to generate initial matrices with nonzero rank-one factors is the following: we generate W and H so that each column or each row has a single non-zero entry (whose position is chosen at random). This leads to four possible initializations denoted SPARSEij: $i = 0$ (resp. $j = 0$) means that W (resp. H) has a single non-zero entry by row, and $i = 1$ (resp. $j = 1$) means that W (resp. H) has a single non-zero entry by column.

Table 3 reports the numerical results. We refer the reader to Sect. 3.3 for the local search NMF algorithm selected. As explained above, RNDCUBE does not perform as well as the sparse initialization strategies (for example, it is not able to find an exact NMF of LEDM8 while all other initialization strategies are). *SPARSE11 has on average the best results* and we will therefore select it as the initialization strategy for MS2 for the remainder of the paper.

Remark 3 It may be surprising that for some symmetric matrices, SPARSE01 works significantly better than SPARSE10 (for example, for LEDM6 and LEDM8) since those two cases are readily seen to be equivalent via transposition (and the corresponding initial errors will be the same in expectation). Symmetry is actually broken because the NMF algorithm used (namely, A-HALS; see Sect. 3.3) first optimizes the factor W (one entry at a time, each entry is optimized several times) while keeping H fixed (in the implementation provided by the authors). Note that, to make sure our observation did not depend on the insufficient number of experiments, we have rerun the same experiment with 1000 initializations for LEDM6 (resp. LEDM8) and sparse01 found 702 (resp. 120) exact NMF's out of these 1000 initializations, while sparse10 found only 83 (resp. 14).

Interestingly, this also explains why SPARSE01 and SPARSE11 (resp. SPARSE00 and SPARSE10) behave similarly: since the first matrix optimized by A-HALS is W, the initial matrix H has more influence on the behaviour of A-HALS than the other initial factor.

Remark 4 Note that more sophisticated initialization strategies have been proposed for NMF, see e.g., [6,56]. However these strategies are usually deterministic, hence cannot provide multiple starts for our heuristics. Still, we have tested the strategies from [6,56] on matrices from Table 1 and observed that they performed quite poorly, failing in most cases to compute an exact NMF.

3.3 Selecting an NMF algorithm

In order to design heuristics for exact NMF, a local search heuristic is needed to improve a given solution (i.e. pair of factors W and H) locally. Most NMF algorithms could potentially be used: in fact, most NMF algorithms are local search heuristics based on standard nonlinear optimization schemes. In this section, we compare the following state-of-the-art NMF algorithms in order to assess their performances for computing exact NMF's:

1. (**MU**) The multiplicative updates (MU) algorithm of [39,40].
2. (**A-MU**) The accelerated MU from [22].
3. (**HALS**) The hierarchical alternating least squares (HALS) algorithm from [10,11].
4. (**A-HALS**) The accelerated HALS from [22].
5. (**ANLS**) The alternating nonnegative least squares algorithm of [38], which alternatively optimizes W and H exactly using a block-pivot active set method; see also [37].

The code of the first four algorithms is available at https://sites.google.com/site/nicolasgillis/. The code of ANLS was obtained from http://www.cc.gatech.edu/~hpark/.

The convergence speeds of these NMF algorithms were previously compared on real-world image and document data sets, and A-HALS was shown to perform the best in most cases. However, in this paper, we are interested in finding exact NMF's of relatively small matrices. Our goal in this paragraph is therefore to identify which algorithm is the best at identifying exact NMF's of such matrices when used as a subroutine for MS2; see Table 4. HALS and A-HALS perform on average the best in terms of number of exact NMF's found (note that A-HALS is not much faster than HALS because the parameter Δt was set to a rather large value, hence both algorithms are able to converge within the allowed time). ANLS performs rather poorly because it runs into numerical problems for rank-deficient factors W (and/or H), which appear as solutions of exact NMF's of nonnegative matrices X with $\text{rank}_+(X) > \text{rank}(X)$ [23]. MU and A-MU also perform poorly: because of their multiplicative nature, they cannot deal very well with sparse solutions;[3] see, e.g., the discussion in [22].

In light of these results, *we select A-HALS* as the NMF algorithm for the remainder of the paper.

[3] Note that we used the variants of MU and A-MU proposed [22] where zero entries of W and H are replaced with a small positive number (we used 10^{-16}) so that they can modify zero entries, and a subsequence is guaranteed to converge to a stationary point [50].

Table 4 Comparison of NMF algorithms combined with multi-start 2

	ANLS	MU	A-MU	HALS	A-HALS
LEDM6	0/100 (∼)	0/100 (∼)	0/100 (∼)	**8/10** (2.8)	7/10 (3.2)
LEDM8	0/100 (∼)	0/100 (∼)	0/100 (∼)	**5/30** (20.8)	**6/40** (20)
LEDM12	0/100 (∼)	0/100 (∼)	0/100 (∼)	0/100 (∼)	0/100 (∼)
LEDM16	0/100 (∼)	0/100 (∼)	0/100 (∼)	0/100 (∼)	0/100 (∼)
LEDM32	0/100 (∼)	0/100 (∼)	0/100 (∼)	0/100 (∼)	0/100 (∼)
6-G	0/100 (∼)	0/100 (∼)	0/100 (∼)	**10/10** (2.1)	**10/10** (1.8)
7-G	0/100 (∼)	0/100 (∼)	0/100 (∼)	**10/10** (2.1)	**10/10** (2.2)
8-G	0/100 (∼)	0/100 (∼)	0/100 (∼)	**9/10** (2.4)	**9/10** (2.3)
9-G	0/100 (∼)	1/100 (405.3)	5/70 (134.2)	5/10 (5.4)	**6/10** (4.2)
16-G	0/100 (∼)	0/100 (∼)	0/100 (∼)	0/100 (∼)	0/100 (∼)
32-G	0/100 (∼)	0/100 (∼)	0/100 (∼)	0/100 (∼)	0/100 (∼)
20-D	0/100 (∼)	0/100 (∼)	0/100 (∼)	0/100 (∼)	0/100 (∼)
24-C	0/100 (∼)	0/100 (∼)	0/100 (∼)	0/100 (∼)	0/100 (∼)
UDISJ4	**10/10** (13)	0/100 (∼)	0/100 (∼)	**10/10** (2.4)	**10/10** (2.1)
UDISJ5	5/100 (778.7)	0/100 (∼)	0/100 (∼)	**5/40** (58.5)	**5/30** (36.4)
UDISJ6	0/100 (∼)	0/100 (∼)	0/100 (∼)	**1/100** (1105.4)	0/100 (∼)
RND1	5/20 (71.6)	3/100 (472.5)	5/10 (33.2)	**10/10** (2.8)	**10/10** (2.3)
RND3	**10/10** (27.5)	8/20 (161.6)	7/10 (49.1)	**10/10** (2.8)	**10/10** (2.4)

4 Two heuristics for NMF

In this section, we propose two heuristics for NMF, along with a hybridization strategy.

4.1 Simulated annealing

The first heuristic we propose follows the widely used simulated annealing framework [45]; see Algorithm SA which we briefly describe here. As for the multi-start heuristics, SA first generates an initial solution (W, H). SA will then explore the neighborhood of this initial solution in a random fashion in the hope to find a better solution. A solution in the neighborhood will be computed by repeating K times the following steps:

– select a small subset $\mathcal{J}$ of J rank-one factors $W(:, \mathcal{J})H(\mathcal{J}, :)$ at random, that is, generate randomly $\mathcal{J} \subset \{1, 2, \dots, r\}$ such that $|\mathcal{J}| = J$,
– reinitialize these rank-one factors randomly (see Sect. 3.2),
– improve the corresponding solution locally (we will use N iterations of A-HALS; see Sect. 3.3), and
– decide whether to keep the refined neighboring solution depending on its error and on the current temperature, see step 14 of Algorithm SA (the higher the temperature, the more likely it is for a solution to be accepted as the next iterate). Note that a solution whose error is smaller than the error of the current solution is always kept. Hence an important characteristic of SA is that it allows for solutions with higher errors to be explored (although the probability for this to happen goes to zero as the temperature decreases).

The procedure is repeated several times for several temperatures (from T_0 to T_{end} with 20 logarithmically-spaced intermediate values). We use the following values for the parameters: initialization SPARSE10, $T_0 = 0.1$ for the initial temperature (this means for example that the initial temperature allows for a solution with relative error 10% higher than the current solution to be accepted with probability $\exp(-1) \approx 0.368$, $T_{end} = 10^{-4}$ for the final temperature (this means for example that the final temperature allows for a solution with relative error 0.1% higher than the current solution to be accepted with probability $\exp(-1)$), $J = 2$, $N = 100$ and $K = 50$; see "Appendix 2" for numerical experiments for different values of the parameters. Note that the choice of these parameters was made within a reasonable range of values. For example, J should not be taken too large: otherwise, it will incur too much change in the factorization computed so far, and will therefore destroy most of the information gained from the previous iterations. Temperatures T_0 and T_{end} should be chosen such that the probability to accept a solution with slightly higher objective function is not too small, nor too high. It is important to point out that the initialization procedure is crucial: in fact, SPARSE10 allows to compute exact NMF for all considered matrices while RND-CUBE fails to do so and is much slower in the cases where it computes exact NMF's; see "Appendix 2".

Algorithm SA Simulated Annealing($X, r, \alpha, \Delta t, T_0, T_{end}, \beta, K, N, J$, tol)

```
Input: $X \in \mathbb{R}^{m \times n}_{+}$, $r < \min(m, n)$, $0 < \alpha < 1$, $T_0$, $T_{end}$, $0 < \beta < 1$, $K, N, J$, tol.
1: $(W, H) \leftarrow$ random initialization$(m, n, r)$. % See Sect. 3.2
2: $e \leftarrow \frac{\|X-WH\|_F}{\|S\|_F}$.
3: $e_{\min} \leftarrow e$.
4: $T \leftarrow T_0$
5: while $T > T_{end}$ do
6:   for $i = 1 \to K$ do
7:     $(\tilde{W}, \tilde{H}) \leftarrow (W, H)$.
8:     $\mathcal{J} \leftarrow$ pick randomly $J$ indices in $\{1, 2, \dots, r\}$.
9:     $(\tilde{W}(:, \mathcal{J}), \tilde{H}(\mathcal{J}, :)) \leftarrow$ random initialization$(m, n, J)$.
10:    $[\tilde{W}, \tilde{H}] \leftarrow$ AlgoNMF$(X, \tilde{W}, \tilde{H}, N)$.
11:    $\tilde{e} \leftarrow \frac{\|X-\tilde{W}\tilde{H}\|_F}{\|X\|_F}$.
12:    $\Delta \leftarrow \tilde{e} - e$.
13:    % $U[0, 1]$ is the uniform distribution in $[0, 1]$ (rand in Matlab)
14:    if $U[0, 1] < \exp\left(-\frac{\Delta}{T}\right)$ then
15:      $W \leftarrow \tilde{W}, H \leftarrow \tilde{H}, e \leftarrow \tilde{e}$.
16:      if $e < e_{\min}$ then
17:        $e_{\min} \leftarrow e$.
18:        $(W_{\min}, H_{\min}) \leftarrow (\tilde{W}, \tilde{H})$.
19:      end if
20:      if $e_{\min} <$ tol then
21:        $T = T_{end}$; break.
22:      end if
23:    end if
24:  end for
25:  $T \leftarrow \beta T$.
26: end while
27: Return $[W_{\min}, H_{\min}]$.
```

4.2 Rank-by-rank heuristic

The second heuristic tries to construct recursively an exact NMF (W, H) of X as follows (see Algorithm RBR):

- at the first step ($k = 1$), an optimal rank-one NMF (W_1, H_1) of X is computed. This can be done for example using the truncated singular value decomposition using the Perron-Frobenius and Eckart-Young theorems.
- At the kth step ($2 \leq k \leq r$), a rank-k NMF solution is generated combining the rank-$(k-1)$ NMF solution (W_{k-1}, H_{k-1}) computed at the $(k-1)$th step with an additional rank-one factor randomly generated. This solution is then locally improved using N steps of an NMF algorithm. This procedure is repeated K times and the best solution is kept; see Algorithm getRankPlusOne.

RBR will turn out to be a powerful exact NMF heuristic for some classes of matrices (such as slack matrices).

Algorithm RBR Rank-by-Rank Heuristic$(X, r, \alpha, \Delta t, K, N)$

```
Input: $X \in \mathbb{R}_+^{m\times n}$, $r < \min(m, n)$, $0 < \alpha < 1$, $\Delta t$, $K$, $N$.
1: $[w_1, \sigma_1, h_1] \leftarrow$ svds$(X, 1)$. % See svds function of Matlab
2: $(W_1, H_1) \leftarrow \left(|w_1|, \sigma_1 \left|h_1^T\right|\right)$ % This is an optimal nonnegative rank-one approximation of X
3: for $k = 2 \to r$ do
4:     $[W_k, H_k] \leftarrow$ getRankPlusOne$(X, W_{k-1}, H_{k-1}, K, N)$.
5: end for
```

Algorithm getRankPlusOne getRankPlusOne(X, W, H, K, N)

```
Input: $X \in \mathbb{R}_+^{m\times n}$, $W \in \mathbb{R}_+^{m\times k-1}$, $H \in \mathbb{R}_+^{k-1\times n}$, $K$, $N$.
1: $e_{\min} \leftarrow 1$.
2: for $j = 1 \to K$ do
3:     $\left(\tilde{W}(:, 1:k-1), \tilde{H}(1:k-1, :)\right) \leftarrow (W, H)$.
4:     $\left(\tilde{W}(:, k), \tilde{H}(k, :)\right) \leftarrow$ random initialization$(m, n, 1)$.
5:     $[\tilde{W}, \tilde{H}] \leftarrow$ AlgoNMF$(X, \tilde{W}, \tilde{H}, N)$.
6:     $\tilde{e} \leftarrow \frac{\|X - \tilde{W}\tilde{H}\|_F}{\|X\|_F}$
7:     if $\tilde{e} < e_{\min}$ then
8:         $e_{\min} \leftarrow \tilde{e}$, $W_{\min} \leftarrow \tilde{W}$, $H_{\min} \leftarrow \tilde{H}$.
9:     end if
10: end for
11: Return $[W_{\min}, H_{\min}]$.
```

We will use SPARSE10 for the initialization, $K = 10$ and $N = 50$ which seem to be a good compromise; see "Appendix 3" for tests of differents values.

4.3 Hybridization

When designing heuristics, a standard technique consists in using hybridization, that is, to combine several heuristics. For example, instead of refining the solution computed by RBR with the final refinement step, it is possible to call Simulated Annealing instead; in other words, we propose to initialize SA with RBR. We refer to this heuristic as ‘Hybrid’.

5 Numerical experiments: comparing exact NMF heuristics

In this section, we compare MS1, MS2, SA, RBR and Hybrid, with a maximimum number of 1000 runs, and stop the execution of an heuristic when 100 exact NMF’s were found (checking this condition every 50 runs); see Table 5.

As already pointed out, the multi-start heuristics perform rather poorly and are not able to compute even a single exact NMF in many cases. We observe that

– RBR is able to compute an exact NMF for all matrices but LEDM32, while SA and Hybrid are able to find an exact NMF for all matrices.
– In terms of robustness, Hybrid is the best as it is able to compute on average the most exact NMF’s for a fixed number of runs.
– In terms of running times, RBR is on average the fastest, while Hybrid is comparatively much slower.

Therefore, in practice, we would recommend to first run RBR as it computes, in many cases, exact NMF’s the fastest. Moreover, for some matrices (e.g., slack matrices of regular n-gons), it is very robust. Then, when RBR fails, we would recommend to run Hybrid because of its robustness: although it is slower, it is in general more likely to find exact NMF’s.

Table 5 Comparison of the different heuristics: MS1 and MS2 (Sect. 3.1), SA (Sect. 4.1), RBR (Sect. 4.2) and Hybrid (Sect. 4.3)

	MS1	MS2	SA	RBR	Hybrid
LEDM6	40/1000 (53.5)	112/150 (3.1)	**100/100** (19.6)	**100/100** (1.4)	**100/100** (19)
LEDM8	0/1000 (∼)	107/600 (27.1)	**100/100** (60.9)	**100/100** (16.7)	**148/150** (63.6)
LEDM12	0/1000 (∼)	0/1000 (∼)	119/200 (42.9)	107/650 (15.1)	**103/150** (36.9)
LEDM16	0/1000 (∼)	0/1000 (∼)	100/250 (118.3)	100/550 (29.1)	**121/250** (104.2)
LEDM32	0/1000 (∼)	0/1000 (∼)	**14/1000** (2592.9)	0/1000 (∼)	**28/1000** (1370.9)
6-G	108/700 (12.1)	**100/100** (2.1)	**100/100** (1.2)	**100/100** (1.4)	**100/100** (1.5)
7-G	104/350 (5.8)	**100/100** (2.2)	**100/100** (4.2)	**100/100** (1.5)	**100/100** (4.4)
8-G	61/1000 (32.2)	129/200 (3.8)	**100/100** (15.4)	**100/100** (1.5)	**100/100** (15.3)
9-G	104/700 (12.8)	117/200 (4.6)	**100/100** (22.9)	**100/100** (1.6)	**100/100** (23.2)
16-G	0/1000 (∼)	0/1000 (∼)	102/350 (91.6)	**143/150** (1.9)	118/150 (34.2)
32-G	0/1000 (∼)	0/1000 (∼)	31/1000 (1086.8)	**107/250** (6.6)	105/300 (97)
20-D	1/1000 (2021.1)	21/1000 (160.9)	**100/100** (7.8)	**129/150** (2.3)	**100/100** (5.6)
24-C	0/1000 (∼)	0/1000 (∼)	**100/100** (3.1)	119/200 (4.1)	**100/100** (4.4)
UDISJ4	102/250 (4)	**100/100** (2.4)	**100/100** (1.2)	**100/100** (1.9)	**100/100** (1.9)
UDISJ5	104/850 (17.4)	102/500 (38)	**100/100** (2.8)	**100/100** (4.9)	**100/100** (5.2)
UDISJ6	7/1000 (337.1)	8/1000 (1594.7)	**100/100** (7.8)	112/450 (66.4)	**100/100** (18.5)
RND1	**148/150** (1.1)	**100/100** (2.8)	**100/100** (1.1)	**100/100** (2.2)	**100/100** (2.2)
RND3	**100/100** (1.1)	**100/100** (2.8)	**100/100** (1.1)	**100/100** (2.2)	**100/100** (2.2)

Table 6 Results for hybrid on larger n-gons

	Hybrid
110-G	**14/1000** (12050.3)
120-G	**12/1000** (15556.4)
130-G	**12/1000** (16462.7)
140-G	**15/1000** (14002.6)
150-G	**5/1000** (49277)
160-G	**1/1000** (144803.1)
170-G	0/1000 ($\sim$)

5.1 Limits of the heuristics for exact NMF

Computing exact NMF's becomes more challenging when the dimensions and the nonnegative rank of the matrix increases, as the computational complexity of the problem depends on these dimensions (see the discussion in Sect. 1.2). To illustrate the limitations of the use of heuristics to find exact NMF's of larger matrices, Table 6 reports the computational results for larger slack matrices of regular n-gons, with the factorization rank provided by conjecture 3.

Moreover, for LEDM of size 48-by-48 or larger, and for slack matrices of regular n-gons with $n \geq 170$, none of our heuristics is able to find a single exact NMF's out of 1000 runs.

6 Using exact NMF heuristics for new insights on the nonnegative rank

In this section, we describe four important open questions related to the nonnegative rank, and show how computing exact NMF's of small matrices can help gain insights about them.

6.1 Kronecker product of two nonnegative matrices

In a recent Dagstuhl seminar [3], participants came up with the following conjecture: given two nonnegative matrices A and B, the nonnegative rank of their Kronecker product is equal to the product of their nonnegative rank, that is,

$$\operatorname{rank}_+(A \otimes B) = \operatorname{rank}_+(A)\, \operatorname{rank}_+(B).$$

Note that this results holds for the usual rank, and that it is easy to show that $\operatorname{rank}_+(A \otimes B) \leq \operatorname{rank}_+(A)\, \operatorname{rank}_+(B)$ (see also [54] for a short discussion). Hamza Fawzi used the multi-start strategy MS1 to come up with the following counter example:

$$A = \begin{pmatrix} 1 & 0 & 1 & a \\ 0 & 1 & 0 & 1-a \\ 0 & 0 & 1 & 1-a \\ 1 & 1 & 0 & a \end{pmatrix},$$

where $a = 3/8$ from [5] for which $\operatorname{rank}_+(A) = 4$ and $\operatorname{rank}_+(A \otimes A) \leq 15$. One may therefore wonder whether the following is true

$$\operatorname{rank}_+(A \otimes B) \geq \operatorname{rank}_+(A)\, \operatorname{rank}_+(B) - 1 \quad ?$$

It turns out that it is also incorrect. In fact, we have found a 4-by-4 nonnegative matrix A with rank 3 and nonnegative rank 4 such that $\text{rank}_+(A \otimes A) = 12$:

$$A = \begin{pmatrix} 1+a & 1-a & 1-a & 1+a \\ 1-a & 1+a & 1+a & 1-a \\ 1+a & 1+a & 1-a & 1-a \\ 1-a & 1-a & 1+a & 1+a \end{pmatrix},$$

where $a = \sqrt{2} - 0.9$. Geometrically, the matrix A is the generalized slack matrix of a pair of polytopes,[4] namely two nested squares: the rows of A correspond to the edges of the outer square and its columns to the vertices of the inner square; see [19] for more details. For $\sqrt{2} - 1 < a \leq 1$, $\text{rank}_+(A) = 4$. However, for $a = 1$ (which corresponds to the slack matrix of the square, that is, the regular 4-gon), $\text{rank}_+(A \otimes A) = 16$. Decreasing a sufficiently while keeping $a > \sqrt{2} - 1$ allows to decrease $\text{rank}_+(A \otimes A)$ to 12 while keeping $\text{rank}_+(A) = 4$. The intuition behind this example is the following: decreasing a leaves more space between the two squares although no triangle fits between the two (hence $\text{rank}_+(A) = 4$). However, this makes the search space of the exact NMF problem for $A \otimes A$ much larger, leading to the existence of an exact NMF with smaller rank.

How the nonnegative rank of the Kronecker product between two matrices relates with their nonnegative ranks remains an open question. This is an important open question, and, as illustrated above, exact NMF algorithms are useful tools to address such questions. In light of the above example, a new conjecture could be the following:

Conjecture 1 *For any nonnegative matrix A,*

$$\text{rank}_+(A \otimes A) \geq \text{rank}_+(A)\, \text{rank}(A).$$

6.2 Slack matrices of regular *n*-gons

As explained in the introduction, the nonnegative rank of the slack matrix X_n of the regular n-gon is equal to its extension complexity, that is, to the minimum number of facets a higher dimensional polytope requires to represent it after a linear projection. It can be shown that $\text{rank}_+(X_n) \geq \lceil \log_2(2n+2) \rceil$ [25]. Ben-Tal and Nemirovski [4] gave an extension of regular n-gons when n is a power of two ($n = 2^k$ for some k) with $2\log_2(n) + 4$ facets. They used this construction to approximate the circle with regular n-gons which allowed them to approximate second-order cone programs with linear programs. Another construction for arbitrary n was proposed in [18] showing that $\text{rank}_+(X_n) \leq 2\lceil \log_2(n) \rceil$. However, the exact value of $\text{rank}_+(X_n)$ is unknown (except for small n; see below).

We have run the Hybrid heuristic on these matrices for all $n \leq 78$ and observe the following:

Conjecture 2 *The nonnegative rank of the slack matrix X_n of the regular n-gon is given by*

$$\text{rank}_+(X_n) = \begin{cases} 2k-1 & \textit{for} \quad 2^{k-1} < n \leq 2^{k-1} + 2^{k-2}, \\ 2k & \textit{for} \quad 2^{k-1} + 2^{k-2} < n \leq 2^k. \end{cases}$$

Note that the conjecture is known to be true for $n \leq 9$ as it matches a lower bound based on the rectangle covering bound improved with additional rank constraints from [43].

[4] The generalized slack matrix of a pair of polytopes P (inner) and Q (outer) is defined as $S(i,j) = b_i - a_i^T v_j$ where $\{x | b_i - a_i^T x \geq 0\}$ is the inequality defining the ith facet of Q and v_j is the jth vertex of P; see, e.g., [29]. Note that the standard slack matrix corresponds to the particular case of equal inner and outer polytopes.

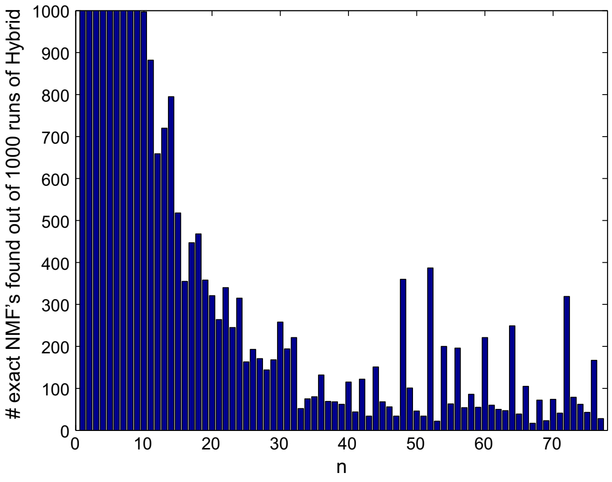

Fig. 2 Number of exact NMF found out of 1000 runs of Hybrid on regular n-gons and for the nonnegative rank given in Conjecture 2

For all slack matrices with[5] $3 \leq n \leq 78$, Hybrid was able to compute at least one exact NMF matching the nonnegative rank given in Conjecture 2, while it was never able to compute a single exact NMF with a smaller nonnegative rank (out of 1000 runs). Figure 2 displays the number of exact NMF's found out of 1000 initializations of Hybrid for the nonnegative rank given in Conjecture 2.

It is interesting and quite natural to observe that, as n increases, Hybrid meets more and more difficulty to compute an exact NMF of these slack matrices. This illustrates the limits of heuristics to solve exact NMF problems for larger (and difficult) matrices; see also Sect. 5.1.

Observing the structure of the factorizations provided by our heuristics, authors in [52] were recently able to provide a formal proof of an upper bound matching the nonnegative rank of Conjecture 2. They also provide a matching lower bound for values of n less or equal than 14 and between 21 and 24, which establishes the conjecture for those values. Conjecture 2 remains open for all other values of n. This reinforces our claim that heuristics for exact NMF are in fact very useful to prove/disprove conjectures on the nonnegative rank.

6.3 Generic n-gons

A n-gon is generic if the coordinates of its vertices are distinct and form a set that is algebraically independent over the rationals; see [18] for more details. It is known that the nonnegative rank of the slack matrices X_n of generic n-gons satisfies

$$\sqrt{2n} \quad \leq \quad \operatorname{rank}_+(X_n) \quad \leq \quad \left\lceil \frac{6n}{7} \right\rceil.$$

The lower bound is due to [18], while the upper bound applies to any n-gon and is due to Shitov [48] (it was also proved using different arguments in [44]). An important question is to characterize the growth of the nonnegative rank of these slack matrices: is it proportional to $\sqrt{n}$, n or something in between [26]? Actually very recent work by Shitov establishes

[5] Because it requires a rather high computational cost for larger n, we stopped testing the conjecture at $n = 78$. In fact, running this experiment on a regular laptop took about two weeks.

sublinearity of the extension complexity of polygons [47] (albeit with a rate that is extremely close to linear).

As n increases, it becomes more and more difficult to generate generic n-gons (because it is likely that a newly generated point belongs to the convex hull of the previously generated points).

Therefore we used the following procedure. We generate random n-gons whose vertices lie on the unit circle. To obtain polygons whose vertices are relatively well separated form the convex hull generated by the other vertices,[6] we subdivide the circle into n disjoint arcs of the same length. Then, each arc is divided into four parts of the same length and we only generate one point randomly into the two middle parts (uniformly distributed). This ensures the angles between any two data points to be larger than $\frac{\pi}{n}$. Then, for each n, we generate ten such random n-gons and run Hybrid with 1000 runs. Table 7 reports the minimum and maximum number of exact NMF's found among these ten matrices.

These results suggest for example that generic 12-gons have extension complexity equal to 9 – which also suggests that all 12-gons have extension complexity smaller than or equal to 9. More generally, these results lead us to the following conjecture

Conjecture 3 *The nonnegative rank of the slack matrix S_n of any n-gon is bounded above by $\left\lfloor \frac{n+6}{2} \right\rfloor$ where $\lfloor x \rfloor$ is the largest integer smaller than x, that is,*

$$\text{rank}_+(S_n) \leq \left\lfloor \frac{n+6}{2} \right\rfloor,$$

and equality holds for $5 \leq n \leq 15$.

Another open question is the following: *For n fixed, are the nonnegative ranks of the slack matrices of all generic n-gons equal to one another?* These experiments suggest that the answer is positive for $n \leq 15$: in fact, in all cases we observe that either no exact NMF is found for the ten randomly generated matrices, or at least some are found for all of them. For $n = 16$, it is less clear whether this is true: we were only able to compute a rank-10 exact NMF for two of the generated matrices. This might be because these matrices are not fully generic, or because, for $n \geq 16$, generic n-gons might have different extension complexities, or because our heuristic fails to compute the exact NMF of such instances. We leave the investigation of these issues for further research.

The validity of conjecture 3 would imply the following.

Corollary (of Conjecture 3) *The nonnegative rank of any rank-3 nonnegative matrix X satisfies*

$$\text{rank}_+(X) \leq \left\lfloor \frac{\min(m,n)+6}{2} \right\rfloor.$$

Sketch of proof This follows from the result in [48]. □

6.4 Extension complexity of the correlation polytope

The convex hull of all $n \times n$ rank-one 0/1 matrices is called the correlation polytope, and we denote its slack matrix $COR(n)$. It was proved in [17] that there exists a positive constant C for which $\text{rank}_+(COR(n)) \geq 2^{Cn}$. This result was improved in [36] where it is shown that $\text{rank}_+(COR(n)) \geq 1.5^n$.

6 As a vertex gets closer and closer to the convex hull generated by the other vertices, it becomes numerically harder and harder to decide whether or not it belongs to the convex hull.

Table 7 Nonnegative rank of random n-gons on the circle: for a given n, the table reports the minimum and maximum number of exact NMF's found by Hybrid out of 1000 runs on ten such n-gons

R/n	6	7	8	9	10	11	12	13	14	15	16	17
4												
5	[0,0]	[0,0]										
6	[1000,1000]	[463,1000]	[0,0]	[0,0]								
7			[754,897]	[160,353]	[0,0]	[0,0]						
8				[743,873]	[351,443]	[25,48]	[0,0]	[0,0]				
9					[787,858]	[401,546]	[148,190]	[10,19]	[0,0]	[0,0]	[0,0]	
10						[692,862]	[580,665]	[242,389]	[63,111]	[5,19]	[0,1]	[0,0]
11							[833,902]	[533,726]	[385,540]	[150,247]	[9,82]	[5,14]
12								[734,874]	[643,766]	[442,631]	[138,365]	[107,204]
13										[671,824]	[375,674]	[405,517]
14											[610,830]	[583,734]
15												[721,829]

Let us define the following $2^n \times 2^n$ matrix, a special instance of UDISJ matrices (see Sect. 2), for which rows and columns are indexed by vectors $a, b \in \{0, 1\}^n$ and such that

$$M_n(a, b) = \left(1 - a^T b\right)^2 .$$

The matrix M_n is a submatrix of the slack matrix of the correlation polytope [17]. For $n = 3, 4, 5, 6$, Hybrid was not able to compute any exact NMF with $r = 2^n - 1$ after 1000 runs. This suggests the following conjecture.

Conjecture 4 *The submatrix M_n of the slack matrix of the correlation polytope has full nonnegative rank, that is,*

$$\text{rank}_+(M_n) = 2^n.$$

This would imply that $\text{rank}_+(COR(n)) \geq 2^n$.

(Note that the rank of M_n is equal to $\frac{n(n+1)}{2} + 1$ for $n \leq 11$. For higher n, the matrix is too large to fit in memory.)

7 Conclusion and further research

We have proposed two new heuristics along with a hybridization for exact nonnegative matrix factorization, and demonstrated that they outperform simpler multi-start strategies when benchmarked on a variety of nonnegative matrices relevant for applications. On the way we proposed a novel efficient initialization strategy, and observed that HALS and A-HALS were suitable as local NMF algorithms when performing exact NMF.

Future research includes the development of new and more efficient heuristics. Also, heuristics can be sensitive to their parameters, especially for matrices for which it is difficult to compute an exact NMF. Hence potential future work also includes fine-tuning the parameters depending on the problem at hand (size of the matrix, difficulty of the corresponding NMF problem, etc.).

The heuristics presented here can readily be applied to find good local minima for the approximate NMF problem (that is, to compute $WH \approx X$), which is particularly useful for real-world applications such as document classification and hyperspectral unmixing. Therefore, it would be an interesting direction for further research to fine-tune and compare heuristics in this context.

So far, we have tested our algorithms on a limited number of nonnegative matrices. It would be good in the future to have a larger library of nonnegative matrices at our disposal, in order to better understand the behavior of the heuristics. With that goal in mind, we will keep our library updated on https://sites.google.com/site/exactnmf and welcome submission of nonnegative matrices, especially those for which computing an exact factorization is still a challenge.

Finally, it is important to recall that, strictly speaking, factorizations presented in this paper were not exact, because they were only computed up to machine precision; see Remark 2. It would therefore also be useful to develop some rounding strategies to transform a high accuracy solution (e.g., 10^{-16} precision) into an exact NMF, when possible. (This was for example done manually for the example of Sect. 6.1 where $\text{rank}_+(A) = 4$ and $\text{rank}_+(A \otimes A) = 12$.)

Acknowledgments The authors would like to thank the reviewers and the editor for their insightful comments which helped improve the paper.

Appendix: Sensitivity to the parameters α and Δt

In this section, we show some numerical results to stress out that the heuristics are not too sensitive (in terms of number of exact NMF's found) to the parameters α and Δt of the local search heuristic (Algorithm FR), as long as they are chosen sufficiently large; see Tables 8 and 9. This is the reason why we selected the rather conservative values of $\alpha = 0.99$ and $\Delta t = 1$ in this paper.

In practice however, it would be good to start the heuristics with smaller values for α and Δt and increase them progressively if the heuristic fails to identify exact NMF's: for easily factorizable matrices (such as the randomly generated ones) it does not make sense to choose large parameters, while for difficult matrices choosing α and Δt too small does not allow the heuristics to find exact NMF's because convergence of NMF algorithms can, in some cases, be too slow.

Parameters for simulated annealing

Table 10 shows the performance of SA for different initialization strategies described in Sect. 3.2 (for $T_0 = 0.1$, $T_{end} = 10^{-4}$, $J = 2$, $N = 100$ and $K = 50$): it appears that SPARSE10 works on average the best hence we keep this initialization for SA. In particular, it is interesting to notice that SPARSE10 is able to compute exact NMF's of 32-G while the other initializations have much more difficulties (only SPARSE00 finds one exact NMF).

Table 11 shows the performance for different values of T_{end} (for $J = 2$, $N = 100$ and $K = 50$): it appears that the value $T_{end} = 10^{-4}$ for the final temperature works well.

Table 8 Comparison of different values of α with $\Delta t = 1$ combined with multi-start 2

	$\alpha = 0.9999$	$\alpha = 0.99$	$\alpha = 0.9$	$\alpha = 0.5$
LEDM6	**8/10** (2.6)	7/10 (3.2)	**8/10** (2.8)	**8/10** ($\underline{2.1}$)
LEDM8	**5/30** (28.8)	**6/40** (20)	**5/30** ($\underline{17.1}$)	**5/30** ($\underline{16.1}$)
LEDM12	0/100 (~)	0/100 (~)	0/100 (~)	0/100 (~)
LEDM16	0/100 (~)	0/100 (~)	0/100 (~)	0/100 (~)
LEDM32	0/100 (~)	0/100 (~)	0/100 (~)	0/100 (~)
6-G	**10/10** ($\underline{1.7}$)	**10/10** ($\underline{1.8}$)	**10/10** (2.1)	**10/10** (2)
7-G	**10/10** ($\underline{1.8}$)	**10/10** (2.2)	**10/10** (2.2)	**10/10** (2.1)
8-G	**9/10** ($\underline{2.4}$)	**9/10** ($\underline{2.3}$)	**9/10** ($\underline{2.3}$)	**9/10** ($\underline{2.3}$)
9-G	5/10 (4.7)	**6/10** ($\underline{4.2}$)	5/10 (4.8)	5/10 (4.7)
16-G	0/100 (~)	0/100 (~)	0/100 (~)	0/100 (~)
32-G	0/100 (~)	0/100 (~)	0/100 (~)	0/100 (~)
20-D	0/100 (~)	0/100 (~)	0/100 (~)	0/100 (~)
24-C	0/100 (~)	0/100 (~)	0/100 (~)	0/100 (~)
UDISJ4	**10/10** (2.4)	**10/10** ($\underline{2.1}$)	**10/10** (2.4)	**10/10** (2.4)
UDISJ5	**6/20** ($\underline{23.1}$)	5/30 (36.4)	6/40 (40.6)	3/100 (179.3)
UDISJ6	0/100 (~)	0/100 (~)	0/100 (~)	0/100 (~)
RND1	**10/10** ($\underline{2.2}$)	**10/10** ($\underline{2.3}$)	**10/10** (2.6)	**10/10** (2.5)
RND3	**10/10** (2.7)	**10/10** ($\underline{2.4}$)	**10/10** ($\underline{2.2}$)	**10/10** (2.7)

Table 9 Comparison of different values of Δt with $\alpha = 0.99$ combined with multi-start 2

	$\Delta t = 0.001$	$\Delta t = 0.01$	$\Delta t = 0.05$	$\Delta t = 0.1$	$\Delta t = 1$	$\Delta t = 2$
LEDM6	**8/10** (1.5)	7/10 (1.7)	**8/10** (1.5)	**8/10** (1.5)	7/10 (3.2)	**8/10** (4.4)
LEDM8	5/50 (12.6)	5/50 (13)	**5/30** (8.6)	**5/30** (9.8)	**6/40** (20)	**5/30** (33.4)
LEDM12	0/100 ($\sim$)	0/100 ($\sim$)	0/100 ($\sim$)	0/100 ($\sim$)	0/100 ($\sim$)	0/100 ($\sim$)
LEDM16	0/100 ($\sim$)	0/100 ($\sim$)	0/100 ($\sim$)	0/100 ($\sim$)	0/100 ($\sim$)	0/100 ($\sim$)
LEDM32	0/100 ($\sim$)	0/100 ($\sim$)	0/100 ($\sim$)	0/100 ($\sim$)	0/100 ($\sim$)	0/100 ($\sim$)
6-G	**10/10** (1.2)	**10/10** (1.1)	**10/10** (1.1)	**10/10** (1.2)	**10/10** (1.8)	**10/10** (3.1)
7-G	**10/10** (1.2)	**10/10** (1.2)	**10/10** (1.2)	**10/10** (1.3)	**10/10** (2.2)	**10/10** (3.1)
8-G	**9/10** (1.4)	**9/10** (1.4)	**9/10** (1.4)	**9/10** (1.4)	**9/10** (2.3)	**9/10** (3.5)
9-G	**6/10** (2.3)	5/10 (2.7)	5/10 (2.7)	5/10 (2.8)	**6/10** (4.2)	5/10 (8.4)
16-G	0/100 ($\sim$)	0/100 ($\sim$)	0/100 ($\sim$)	0/100 ($\sim$)	0/100 ($\sim$)	0/100 ($\sim$)
32-G	0/100 ($\sim$)	0/100 ($\sim$)	0/100 ($\sim$)	0/100 ($\sim$)	0/100 ($\sim$)	0/100 ($\sim$)
20-D	0/100 ($\sim$)	0/100 ($\sim$)	0/100 ($\sim$)	0/100 ($\sim$)	0/100 ($\sim$)	0/100 ($\sim$)
24-C	0/100 ($\sim$)	0/100 ($\sim$)	0/100 ($\sim$)	0/100 ($\sim$)	0/100 ($\sim$)	0/100 ($\sim$)
UDISJ4	**10/10** (1.5)	**10/10** (1.5)	**10/10** (1.5)	**10/10** (1.6)	**10/10** (2.1)	**10/10** (3.4)
UDISJ5	1/100 (606.2)	1/100 (614.9)	4/100 (153.6)	2/100 (306.3)	5/30 (36.4)	**5/20** (37.5)
UDISJ6	0/100 ($\sim$)	0/100 ($\sim$)	0/100 ($\sim$)	0/100 ($\sim$)	0/100 ($\sim$)	0/100 ($\sim$)
RND1	**10/10** (1.9)	**10/10** (1.8)	**10/10** (1.9)	**10/10** (1.9)	**10/10** (2.3)	**10/10** (3.8)
RND3	**10/10** (1.9)	**10/10** (1.8)	**10/10** (1.9)	**10/10** (1.9)	**10/10** (2.4)	**10/10** (3.8)

Table 10 Comparison of the different initialization strategies combined with SA

	Sparse 00	Sparse 10	Sparse 01	Sparse 11	RNDCUBE
LEDM6	**10/10** (19.5)	**10/10** (17)	**10/10** (20.4)	**10/10** (16.5)	**10/10** (19.4)
LEDM8	**10/10** (57.9)	**10/10** (44.8)	**10/10** (59)	**9/10** (49.9)	**10/10** (63.3)
LEDM12	**9/10** (26.5)	6/10 (30)	11/20 (45.1)	8/10 (28)	10/20 (51)
LEDM16	7/20 (125.9)	**11/20** (65.6)	5/10 (99.2)	6/20 (112.8)	6/20 (132.6)
LEDM32	**5/90** (711.7)	**3/100** (1016.9)	**1/100** (3728.4)	**2/100** (1447.5)	0/100 ($\sim$)
6-G	**10/10** (1.2)	**10/10** (1.2)	**10/10** (1.1)	**10/10** (1.3)	**10/10** (1.2)
7-G	**10/10** (3.5)	**10/10** (3.5)	**9/10** (72.5)	**10/10** (3.4)	**10/10** (3.5)
8-G	**10/10** (17.2)	**10/10** (13.8)	**10/10** (19.6)	**10/10** (15.9)	**10/10** (16.1)
9-G	**10/10** (22.7)	**10/10** (21.3)	**10/10** (23)	**10/10** (18)	**10/10** (24.1)
16-G	6/20 (87.6)	**8/20** (61.4)	7/40 (150.4)	5/60 (287.7)	6/40 (182.7)
32-G	0/100 ($\sim$)	**3/100** (999.5)	0/100 ($\sim$)	0/100 ($\sim$)	0/100 ($\sim$)
20-D	**10/10** (7.5)	**10/10** (4.2)	**10/10** (8.1)	**10/10** (10)	**10/10** (8)
24-C	**10/10** (4.4)	**10/10** (3.6)	**10/10** (3.1)	**10/10** (4.5)	**10/10** (3.7)
UDISJ4	**10/10** (1.2)	**10/10** (1.2)	**10/10** (1.1)	**10/10** (1.1)	**10/10** (1.1)
UDISJ5	**10/10** (2.9)	**10/10** (2.3)	**10/10** (3)	**10/10** (2.7)	**10/10** (3.8)
UDISJ6	**10/10** (8.3)	**10/10** (8.1)	**10/10** (52.4)	**10/10** (13.5)	**10/10** (43.9)
RND1	**10/10** (1.1)	**10/10** (1.1)	**10/10** (1.1)	**10/10** (1.1)	**10/10** (1.1)
RND3	**10/10** (1.1)	**10/10** (1.1)	**10/10** (1.1)	**10/10** (1.1)	**10/10** (1.1)

Table 11 Performance of simulated annealing for different values of T_{end} ($J = 2$, $N = 100$ and $K = 50$)

	$T_{end} = 10^{-2}$	$T_{end} = 10^{-3}$	$T_{end} = 10^{-4}$	$T_{end} = 10^{-5}$	$T_{end} = 10^{-6}$
LEDM6	**10/10** (17.8)	**10/10** (19)	**10/10** (17)	**10/10** ($\underline{12.3}$)	**10/10** (13.6)
LEDM8	**10/10** (54.5)	**10/10** (57)	**10/10** ($\underline{44.8}$)	**10/10** (62.3)	**9/10** ($\underline{49.1}$)
LEDM12	6/60 (225.4)	6/20 (63.9)	6/10 ($\underline{30}$)	5/10 (33)	**7/10** ($\underline{29.4}$)
LEDM16	5/100 (488.7)	5/50 (223)	**11/20** ($\underline{65.6}$)	5/10 (84.1)	6/20 (100.1)
LEDM32	0/100 ($\sim$)	0/100 ($\sim$)	**3/100** ($\underline{1016.9}$)	0/100 ($\sim$)	**2/100** (1561)
6-G	**10/10** ($\underline{1.2}$)	**10/10** ($\underline{1.2}$)	**10/10** ($\underline{1.2}$)	**10/10** ($\underline{1.2}$)	**10/10** ($\underline{1.2}$)
7-G	**10/10** (3.9)	**10/10** ($\underline{3.8}$)	**10/10** ($\underline{3.5}$)	**10/10** (3.9)	**10/10** (4.2)
8-G	**10/10** (16.5)	**10/10** (17.4)	**10/10** (13.8)	**10/10** ($\underline{11.4}$)	**10/10** (13.4)
9-G	**10/10** (21)	**10/10** (17.2)	**10/10** (21.3)	**10/10** ($\underline{12.7}$)	**10/10** (15.8)
16-G	5/80 (352.4)	6/20 (66.3)	8/20 (61.4)	6/30 (97.4)	**6/10** ($\underline{23.7}$)
32-G	0/100 ($\sim$)	0/100 ($\sim$)	**3/100** ($\underline{999.5}$)	**3/100** ($\underline{931.3}$)	**2/100** (1331.3)
20-D	**10/10** ($\underline{4.1}$)	**10/10** (5.9)	**10/10** ($\underline{4.2}$)	**10/10** (7.6)	**10/10** (5.2)
24-C	**10/10** (3.9)	**10/10** ($\underline{2.1}$)	**10/10** (3.6)	**10/10** (2.9)	**10/10** (2.7)
UDISJ4	**10/10** ($\underline{1.2}$)	**10/10** ($\underline{1.2}$)	**10/10** ($\underline{1.2}$)	**10/10** ($\underline{1.1}$)	**10/10** ($\underline{1.2}$)
UDISJ5	**10/10** ($\underline{2.3}$)	**10/10** ($\underline{2.4}$)	**10/10** ($\underline{2.3}$)	**10/10** ($\underline{2.5}$)	**10/10** ($\underline{2.3}$)
UDISJ6	**10/10** (9)	**10/10** (9.4)	**10/10** ($\underline{8.1}$)	**10/10** ($\underline{7.7}$)	**10/10** (8.6)
RND1	**10/10** ($\underline{1.1}$)	**10/10** ($\underline{1.1}$)	**10/10** ($\underline{1.1}$)	**10/10** ($\underline{1.1}$)	**10/10** ($\underline{1.1}$)
RND3	**10/10** ($\underline{1.1}$)	**10/10** ($\underline{1.1}$)	**10/10** ($\underline{1.1}$)	**10/10** ($\underline{1.1}$)	**10/10** ($\underline{1.1}$)

Table 12 shows the performance for different values of N and K, for $T_{end} = 10^{-4}$ and $J = 2$. It seems that $K = 50$ and $N = 100$ is a good compromise between number of exact NMF's found and computational time.

Table 13 shows the performance for different values of J (for $T_{end} = 10^{-4}$, $K = 50$ and $N = 100$), and shows that $J = 2$ performs the best.

Parameters for the rank-by-rank heuristic

Table 14 shows the performance of RBR for the different initialization strategies (for $N = 100$ and $K = 50$): SPARSE10 works on average the best. As for SA, it allows to compute exact NMF's of 32-G (6/10) while all other initializations fail.

Table 15 gives the results for several values of the parameters K and N. It is interesting to observe that when K gets larger, the heuristic performs rather poorly in some cases (e.g., for the UDISJ6 matrix). The reason is that when K increases, the heuristic tends to generate similar solutions: the ones obtained with Algorithm getRankPlusOne initialized with the best solution that can be obtained by combining the rank-($k - 1$) solution with a rank-one one. In other words, the search domain that can be explored by RBR is reduced when K increases.

Initialization for the hybridization

Again the best initialization strategy is SPARSE10. However, it is interesting to note that Hybrid is less sensitive to initialization than SA and RBR. In fact, except for 32-G with

Table 12 Performance of simulated annealing for different values of K and N ($T_{end} = 10^{-4}$ and $J = 2$)

	$N = 10$	$N = 50$	$N = 100$	$N = 10$	$N = 50$	$N = 100$
	$K = 10$			$K = 20$		
LEDM6	6/10 (3.1)	**10/10** (3.1)	**10/10** (4.9)	8/10 (2.8)	**10/10** (5)	**10/10** (8.7)
LEDM8	7/20 (53.1)	7/10 (45)	8/10 (48)	**9/10** (49)	**10/10** (48)	**10/10** (49.8)
LEDM12	5/80 (47.2)	7/20 (13.2)	5/30 (40.8)	5/30 (21.6)	5/10 (12.7)	6/10 (18.4)
LEDM16	5/70 (98.1)	5/50 (115.4)	5/80 (184.6)	5/30 (76.7)	6/30 (93.9)	5/20 (91.2)
LEDM32	0/100 ($\sim$)	0/100 ($\sim$)	0/100 ($\sim$)	0/100 ($\sim$)	0/100 ($\sim$)	0/100 ($\sim$)
6-G	**10/10** (1.6)	**10/10** (2.5)	**10/10** (1.2)	**10/10** (2)	**10/10** (3.4)	**10/10** (1.2)
7-G	7/10 (2.7)	**10/10** (3)	**10/10** (2.4)	**9/10** (2.4)	**10/10** (4.7)	**10/10** (3.8)
8-G	**10/10** (1.6)	**10/10** (3.1)	**10/10** (4.8)	**10/10** (2.1)	**10/10** (5)	**10/10** (7.3)
9-G	8/10 (2.3)	7/10 (5.1)	**10/10** (5.5)	8/10 (3)	**10/10** (5.5)	**10/10** (9.9)
16-G	5/30 (15.1)	7/60 (38.8)	5/40 (56.5)	5/30 (18.7)	6/20 (21.7)	5/20 (46)
32-G	1/100 (410.7)	0/100 ($\sim$)	1/100 (895.7)	0/100 ($\sim$)	2/100 (458.4)	2/100 (740)
20-D	6/10 (3.5)	8/10 (4.8)	**9/10** (4.9)	5/10 (5.7)	7/10 (9.3)	**9/10** (6.4)
24-C	8/10 (2.5)	**9/10** (4.9)	**10/10** (2.9)	8/10 (3.3)	8/10 (9.6)	**9/10** (4.4)
UDISJ4	**10/10** (1.7)	**10/10** (1.3)	**10/10** (1.2)	**10/10** (2.4)	**10/10** (1.4)	**10/10** (1.2)
UDISJ5	**10/10** (3.9)	**10/10** (13.7)	**10/10** (2.7)	**10/10** (6.8)	**10/10** (26.3)	**10/10** (3.5)
UDISJ6	**9/10** (6.9)	**10/10** (22.9)	**10/10** (7.1)	**10/10** (11.4)	**10/10** (42.3)	**10/10** (8.8)
RND1	**10/10** (1.9)	**10/10** (1.3)	**10/10** (1.1)	**10/10** (2.7)	**10/10** (1.3)	**10/10** (1.1)
RND3	**10/10** (1.9)	**10/10** (1.1)	**10/10** (1.1)	**10/10** (2.7)	**10/10** (1.1)	**10/10** (1.1)
	$K = 50$			$K = 100$		
LEDM6	8/10 (4.6)	**10/10** (11.1)	**10/10** (17)	**10/10** (5.9)	**10/10** (19)	**10/10** (38.4)
LEDM8	**9/10** (44)	**9/10** (59.4)	**10/10** (44.8)	**9/10** (49.4)	**10/10** (62.1)	**10/10** (71.5)
LEDM12	6/20 (15.4)	6/10 (23)	6/10 (30)	5/20 (31.5)	7/10 (33.8)	**9/10** (55.9)
LEDM16	7/60 (103.5)	5/20 (100.9)	11/20 (65.6)	6/40 (116.4)	**7/10** (81.2)	6/10 (156)
LEDM32	0/100 ($\sim$)	**3/100** (706.4)	**3/100** (1016.9)	**1/100** (1318)	**5/70** (571.8)	**5/70** (1049)
6-G	**10/10** (3.5)	**10/10** (2.7)	**10/10** (1.2)	**10/10** (5.4)	**10/10** (3.9)	**10/10** (1.2)
7-G	**10/10** (3.7)	**10/10** (9.3)	**10/10** (3.5)	**10/10** (5.9)	**10/10** (18.2)	**10/10** (4.5)
8-G	**10/10** (3.8)	**10/10** (11)	**10/10** (13.8)	**10/10** (5.9)	**10/10** (21.1)	**10/10** (28)
9-G	**9/10** (4.4)	**10/10** (11.8)	**10/10** (21.3)	**9/10** (7.1)	**10/10** (23.3)	**10/10** (42.4)
16-G	6/20 (16.3)	9/20 (31.5)	8/20 (61.4)	10/20 (14.4)	9/20 (58.7)	**8/10** (63.6)
32-G	2/100 (341.1)	3/100 (606.6)	3/100 (999.5)	3/100 (330.4)	5/60 (405.1)	**5/40** (522.4)
20-D	**10/10** (4.4)	**10/10** (14.3)	**10/10** (4.2)	**9/10** (8.4)	**10/10** (27.7)	**10/10** (11.8)
24-C	**10/10** (5)	**10/10** (17.1)	**10/10** (3.6)	**10/10** (8.7)	**10/10** (30.5)	**10/10** (4.2)
UDISJ4	**10/10** (4.4)	**10/10** (1.3)	**10/10** (1.2)	**10/10** (7.4)	**10/10** (1.4)	**10/10** (1.2)
UDISJ5	**10/10** (15.5)	**10/10** (55.6)	**10/10** (2.3)	**10/10** (28.6)	**10/10** (98.7)	**10/10** (3.5)
UDISJ6	**10/10** (27.4)	**10/10** (63.5)	**10/10** (8.1)	**10/10** (48.5)	**10/10** (131.5)	**10/10** (11.3)
RND1	**10/10** (5.1)	**10/10** (1.3)	**10/10** (1.1)	**10/10** (8.5)	**10/10** (1.3)	**10/10** (1.1)
RND3	**10/10** (5.1)	**10/10** (1.1)	**10/10** (1.1)	**10/10** (8.5)	**10/10** (1.1)	**10/10** (1.1)

Table 13 Performance of simulated annealing for different values of J ($T_{end} = 10^{-4}$, $K = 50$ and $N = 100$)

	$\|J\| = 1$	$\|J\| = 2$	$\|J\| = 3$	$\|J\| = 4$
LEDM6	**10/10** (20.5)	**10/10** (17)	**10/10** (20.3)	**10/10** (19.4)
LEDM8	**10/10** (54.4)	**10/10** (44.8)	**10/10** (64.4)	**10/10** (61.3)
LEDM12	**10/10** (25)	6/10 (30)	5/10 (50.3)	7/20 (67.7)
LEDM16	5/10 (100.9)	**11/20** (65.6)	7/20 (115.1)	6/20 (99.4)
LEDM32	**1/100** (3655.9)	**3/100** (1016.9)	**1/100** (3688)	**2/100** (1798)
6-G	**10/10** (1.2)	**10/10** (1.2)	**10/10** (1.2)	**10/10** (1.2)
7-G	**10/10** (2.2)	**10/10** (3.5)	**10/10** (5.1)	**10/10** (10.4)
8-G	**10/10** (17.5)	**10/10** (13.8)	**10/10** (16.5)	**10/10** (20.4)
9-G	**10/10** (19.5)	**10/10** (21.3)	**10/10** (22.8)	**10/10** (23.5)
16-G	**6/10** (44.5)	8/20 (61.4)	5/20 (108.2)	6/60 (268.2)
32-G	**5/90** (613.1)	**3/100** (999.5)	0/100 (∼)	**1/100** (3377)
20-D	**9/10** (8)	**10/10** (4.2)	**10/10** (8.6)	**10/10** (19.8)
24-C	**10/10** (4)	**10/10** (3.6)	**10/10** (4.8)	**10/10** (6.5)
UDISJ4	**10/10** (1.3)	**10/10** (1.2)	**10/10** (1.2)	**10/10** (1.2)
UDISJ5	**10/10** (3.7)	**10/10** (2.3)	**10/10** (2.8)	**10/10** (3)
UDISJ6	**10/10** (11)	**10/10** (8.1)	**10/10** (6.8)	**10/10** (8.1)
RND1	**10/10** (1.1)	**10/10** (1.1)	**10/10** (1.2)	**10/10** (1.1)
RND3	**10/10** (1.1)	**10/10** (1.1)	**10/10** (1.1)	**10/10** (1.1)

Table 14 Comparison of the different initialization strategies combined with RBR

	Sparse 00	Sparse 10	S01	Sparse 11	RNDCUBE
LEDM6	**10/10** (1.4)	**10/10** (1.4)	**10/10** (1.4)	**10/10** (1.4)	**10/10** (1.4)
LEDM8	**10/10** (14.6)	**10/10** (15.8)	**10/10** (12.3)	**10/10** (20.3)	**10/10** (11.7)
LEDM12	5/30 (14.8)	**7/30** (10.1)	**6/20** (7.8)	**7/30** (10.3)	**5/20** (9.4)
LEDM16	5/50 (40.4)	5/30 (29.5)	6/40 (31.5)	5/30 (29.5)	**6/10** (17.9)
LEDM32	0/100 (∼)	0/100 (∼)	0/100 (∼)	0/100 (∼)	0/100 (∼)
6-G	**10/10** (1.4)	**10/10** (1.4)	**10/10** (1.4)	**10/10** (1.4)	**10/10** (1.4)
7-G	**10/10** (1.5)	**10/10** (1.5)	**10/10** (1.5)	**10/10** (1.5)	**10/10** (1.5)
8-G	**10/10** (1.5)	**10/10** (1.5)	**10/10** (1.5)	**10/10** (1.5)	**10/10** (1.5)
9-G	**10/10** (1.6)	**10/10** (1.6)	**10/10** (1.6)	**10/10** (1.6)	**10/10** (1.6)
16-G	**10/10** (1.8)	**10/10** (1.8)	**9/10** (2.1)	5/10 (4.3)	6/20 (8.1)
32-G	**7/20** (8.4)	**8/20** (7.1)	5/40 (26.7)	**7/20** (9.4)	1/100 (421.1)
20-D	8/10 (2.5)	**9/10** (2.1)	**10/10** (1.9)	**9/10** (2.2)	8/10 (2.3)
24-C	7/10 (3.4)	**8/10** (2.9)	**8/10** (2.9)	**8/10** (3)	12/20 (4.1)
UDISJ4	**10/10** (1.9)	**10/10** (1.9)	**10/10** (1.9)	**10/10** (1.9)	8/10 (2.3)
UDISJ5	**10/10** (5)	**10/10** (4.8)	**10/10** (4.9)	**10/10** (4.9)	**9/10** (5.5)
UDISJ6	6/20 (57.5)	6/40 (116.8)	7/10 (23.7)	**8/10** (21)	5/30 (106)
RND1	**10/10** (2.2)	**10/10** (2.2)	**10/10** (2.2)	**10/10** (2.2)	**10/10** (2.2)
RND3	**10/10** (2.2)	**10/10** (2.2)	**10/10** (2.2)	**10/10** (2.2)	**10/10** (2.2)

Table 15 Performance of the rank-by-rank heuristic for different values of K and N

	$N = 10$	$N = 50$	$N = 100$	$N = 10$	$N = 50$	$N = 100$
	$K = 1$			$K = 10$		
LEDM6	8/30 (6.7)	7/10 (2)	5/10 (3.2)	**10/10** (1.2)	**10/10** (1.4)	**10/10** (1.7)
LEDM8	6/10 (19.4)	8/20 (13.6)	6/10 (9.8)	**10/10** (38.5)	**10/10** (15.8)	**10/10** (13.1)
LEDM12	5/50 (27.8)	5/20 (10.2)	7/20 (6.2)	5/80 (33.8)	7/30 (10.1)	**10/10** (2.2)
LEDM16	0/100 (∼)	5/60 (59.1)	5/40 (37.2)	5/100 (73.8)	5/30 (29.5)	6/40 (36.4)
LEDM32	**1/100** (545.3)	0/100 (∼)	0/100 (∼)	0/100 (∼)	0/100 (∼)	0/100 (∼)
6-G	5/10 (3.1)	5/10 (2.8)	6/20 (4)	**10/10** (1.2)	**10/10** (1.4)	**10/10** (1.7)
7-G	12/20 (2.4)	7/10 (2)	6/10 (2.1)	**10/10** (1.2)	**10/10** (1.5)	**10/10** (1.9)
8-G	**10/10** (1.1)	**10/10** (1.1)	**10/10** (1.2)	**10/10** (1.2)	**10/10** (1.5)	**10/10** (1.9)
9-G	5/10 (3.3)	11/20 (2.8)	6/10 (2.6)	**10/10** (1.2)	**10/10** (1.6)	**10/10** (2.2)
16-G	5/70 (29.7)	6/30 (10.2)	5/50 (24.3)	8/10 (1.8)	**10/10** (1.8)	**10/10** (2.4)
32-G	1/100 (316.6)	2/100 (153.4)	5/100 (61.2)	7/30 (9.8)	8/20 (7.1)	11/20 (6.8)
20-D	7/20 (4.9)	5/30 (11.5)	8/30 (6.4)	**10/10** (1.3)	**9/10** (2.1)	8/10 (3.4)
24-C	5/70 (28.1)	7/20 (3.9)	5/50 (15.1)	6/10 (2.9)	8/10 (2.9)	8/10 (4.6)
UDISJ4	**9/10** (1.3)	**9/10** (1.3)	**9/10** (1.4)	**10/10** (1.3)	**10/10** (1.9)	**9/10** (3)
UDISJ5	8/10 (1.8)	**9/10** (1.7)	6/10 (3.2)	**10/10** (2.1)	**10/10** (4.8)	**10/10** (8.4)
UDISJ6	**7/10** (2.6)	**7/10** (4)	**13/20** (6.9)	7/20 (15.1)	6/40 (116.8)	7/30 (149.7)
RND1	**10/10** (1.1)	**10/10** (1.2)	**10/10** (1.3)	**10/10** (1.3)	**10/10** (2.2)	**10/10** (3.1)
RND3	**10/10** (1.1)	**10/10** (1.2)	**10/10** (1.3)	**10/10** (1.3)	**10/10** (2.2)	**10/10** (3.1)
	$K = 50$			$K = 100$		
LEDM6	**10/10** (1.4)	**10/10** (2.7)	**10/10** (4.1)	**10/10** (1.9)	**10/10** (4.4)	**10/10** (7.7)
LEDM8	**10/10** (20.8)	**10/10** (16)	**10/10** (28.8)	**10/10** (19.2)	**10/10** (16.5)	**10/10** (34.7)
LEDM12	6/50 (21.2)	9/20 (10.1)	**10/10** (6.2)	6/40 (22)	6/10 (11.7)	**10/10** (12.3)
LEDM16	5/50 (44.2)	7/50 (53.2)	5/50 (98.7)	**6/20** (27.1)	5/70 (139.3)	6/70 (197.6)
LEDM32	0/100 (∼)	0/100 (∼)	0/100 (∼)	0/100 (∼)	0/100 (∼)	0/100 (∼)
6-G	**10/10** (1.4)	**10/10** (2.7)	**10/10** (4.1)	**10/10** (1.9)	**10/10** (4.3)	**10/10** (7.6)
7-G	**10/10** (1.6)	**10/10** (3.2)	**10/10** (5)	**10/10** (2.1)	**10/10** (5.4)	**10/10** (9.6)
8-G	**10/10** (1.6)	**10/10** (3.2)	**10/10** (5.1)	**10/10** (2.1)	**10/10** (5.4)	**10/10** (9.7)
9-G	**10/10** (1.7)	**10/10** (3.8)	**10/10** (6.2)	**10/10** (2.4)	**10/10** (6.6)	**10/10** (12)
16-G	**10/10** (1.9)	**10/10** (4.5)	**10/10** (7.5)	**10/10** (2.7)	**10/10** (8)	**10/10** (14.9)
32-G	5/20 (14.4)	7/10 (9.3)	**10/10** (10.8)	6/50 (39.4)	**9/10** (12.8)	**10/10** (21.7)
20-D	**10/10** (2)	8/10 (6.4)	**9/10** (9.7)	**10/10** (3)	**9/10** (10.3)	5/10 (34.7)
24-C	9/20 (6.9)	**10/10** (7.6)	**9/10** (14.7)	7/20 (13.5)	**10/10** (13.8)	**9/10** (29.8)
UDISJ4	**10/10** (2)	**10/10** (5.1)	8/10 (10.9)	**10/10** (3)	**10/10** (9)	7/40 (98.6)
UDISJ5	8/10 (7)	**10/10** (19.9)	**10/10** (35.8)	5/10 (20.4)	**10/10** (38.2)	**10/10** (74.8)
UDISJ6	5/40 (156.2)	0/100 (∼)	0/100 (∼)	5/40 (293.2)	0/100 (∼)	0/100 (∼)
RND1	**10/10** (2.4)	**10/10** (6.4)	**10/10** (12.2)	**10/10** (3.7)	**10/10** (12.4)	**10/10** (22.4)
RND3	**10/10** (2.4)	**10/10** (6.4)	**10/10** (12.3)	**10/10** (3.7)	**10/10** (12.4)	**10/10** (22.3)

Table 16 Comparison of the different initialization strategies combined with the hybridization between RBR and SA

	Sparse 00	Sparse 10	Sparse 01	Sparse 11	RNDCUBE
LEDM6	**10/10** (20.1)	**10/10** (20.4)	**10/10** (20.2)	**10/10** (20)	**10/10** (20.1)
LEDM8	**10/10** (59.7)	**10/10** (59.2)	**10/10** (53)	**10/10** (65.9)	**10/10** (61.6)
LEDM12	7/10 (36)	5/10 (50.8)	5/10 (51)	7/10 (36.7)	**8/10** (31.2)
LEDM16	5/10 (102.6)	8/20 (103.1)	11/20 (91.3)	5/10 (69.8)	**7/10** (74.5)
LEDM32	**4/100** (946.2)	**2/100** (1851.1)	**1/100** (3796.3)	0/100 (∼)	0/100 (∼)
6-G	**10/10** (1.5)	**10/10** (1.4)	**10/10** (1.5)	**10/10** (1.5)	**10/10** (1.4)
7-G	**10/10** (4.5)	**10/10** (3.1)	**10/10** (2.5)	**10/10** (3.5)	**10/10** (3.1)
8-G	**10/10** (14.7)	**10/10** (13.4)	**10/10** (19.4)	**10/10** (20.2)	**10/10** (19.2)
9-G	**10/10** (22.9)	**10/10** (22)	**10/10** (23.8)	**10/10** (24.1)	**10/10** (23.9)
16-G	**10/10** (26.4)	7/10 (36.7)	8/10 (34.6)	6/10 (45.6)	5/20 (109.3)
32-G	**5/10** (67.4)	**5/10** (66.9)	6/30 (176.6)	6/40 (235.9)	0/100 (∼)
20-D	**10/10** (10.1)	**10/10** (4.4)	**10/10** (10.3)	**10/10** (6.7)	**10/10** (5.7)
24-C	**10/10** (2.7)	**10/10** (5.6)	**10/10** (3.1)	**10/10** (4.1)	**10/10** (2.9)
UDISJ4	**10/10** (1.9)	**10/10** (1.9)	**10/10** (1.9)	**10/10** (1.9)	**10/10** (1.9)
UDISJ5	**10/10** (5.1)	**10/10** (5)	**10/10** (5.3)	**10/10** (5.3)	**10/10** (5.8)
UDISJ6	**10/10** (18.6)	**10/10** (21.2)	**10/10** (19.3)	**10/10** (17.2)	**10/10** (21.4)
RND1	**10/10** (2.2)	**10/10** (2.2)	**10/10** (2.3)	**10/10** (2.2)	**10/10** (2.2)
RND3	**10/10** (2.2)	**10/10** (2.2)	**10/10** (2.2)	**10/10** (2.2)	**10/10** (2.2)

RNDCUBE and LEDM32 with SPARSE01, it was able to compute exact NMF's in all situations. In other words, as shown in Table 16, Hybrid is a more robust strategy than RBR and SA although it is computationally more expensive on average.

References

1. Arora, S., Ge, R., Kannan, R., Moitra, A.: Computing a nonnegative matrix factorization—provably. in *Proceedings of the 44th Symposium on Theory of Computing, STOC '12*, pp. 145–162, (2012)
2. Beasley, L., Laffey, T.: Real rank versus nonnegative rank. Linear Algebra Appl. **431**(12), 2330–2335 (2009)
3. Beasley, L., Lee, T., Klauck, H., Theis, D.: Dagstuhl report 13082: communication complexity, linear optimization, and lower bounds for the nonnegative rank of matrices (2013). arXiv:1305.4147
4. Ben-Tal, A., Nemirovski, A.: On polyhedral approximations of the second-order cone. Math. Oper. Res. **26**(2), 193–205 (2001)
5. Bocci, C., Carlini, E., Rapallo, F.: Perturbation of matrices and nonnegative rank with a view toward statistical models. SIAM J. Matrix Anal. Appl. **32**(4), 1500–1512 (2011)
6. Boutsidis, C., Gallopoulos, E.: SVD based initialization: a head start for nonnegative matrix factorization. Pattern Recognit. **41**(4), 1350–1362 (2008)
7. Brown, C.W.: Qepcad b: a program for computing with semi-algebraic sets using cads. ACM SIGSAM Bull. **37**(4), 97–108 (2003)
8. Carlini, E., Rapallo, F.: Probability matrices, non-negative rank, and parameterization of mixture models. Linear Algebra Appl. **433**, 424–432 (2010)
9. Cichocki, A., Amari, S.-I., Zdunek, R., Phan, A.: Non-negative Matrix and Tensor Factorizations: Applications to Exploratory Multi-way Data Analysis and Blind Source Separation. Wiley, London (2009)
10. Cichocki, A., Phan, A.H.: Fast local algorithms for large scale nonnegative matrix and tensor factorizations. IEICE Trans. Fundam. Electron. **E92–A**(3), 708–721 (2009)

11. Cichocki, A., Zdunek, R., Amari, S.-i.: Hierarchical ALS Algorithms for Nonnegative Matrix and 3D Tensor Factorization. Lecture notes in computer science (Springer, 2007), pp. 169–176
12. Cohen, J., Rothblum, U.: Nonnegative ranks, decompositions and factorization of nonnegative matrices. Linear Algebra Appl. **190**, 149–168 (1993)
13. Conforti, M., Cornuéjols, G., Zambelli, G.: Extended formulations in combinatorial optimization. 4OR A Q.J. Oper. Res. **10**(1), 1–48 (2010)
14. de Caen, D., Gregory, D.A., Pullman, N.J.: The boolean rank of zero-one matrices. in *Proceedings of Third Caribbean Conference on Combinatorics and Computing* (Barbados), pp. 169–173 (1981)
15. Fawzi, H., Gouveia, J., Parrilo, P., Robinson, R., Thomas, R.: Positive Semidefinite Rank (2014). arXiv:1407.4095
16. Fiorini, S., Kaibel, V., Pashkovich, K., Theis, D.: Combinatorial bounds on nonnegative rank and extended formulations. Discret. Math. **313**(1), 67–83 (2013)
17. Fiorini, S., Massar, S., Pokutta, S., Tiwary, H., de Wolf, R.: Linear Versus Semidefinite Extended Formulations: Exponential Separation and Strong Lower Bounds. in *Proceedings of the Forty-Fourth Annual ACM Symposium on Theory of Computing*, ACM, pp. 95–106, (2012)
18. Fiorini, S., Rothvoss, T., Tiwary, H.: Extended formulations for polygons. Discret. Comput. Geom. **48**(3), 658–668 (2012)
19. Gillis, N.: Sparse and unique nonnegative matrix factorization through data preprocessing. J. Mach. Learn. Res. **13**(Nov), 3349–3386 (2012)
20. Gillis, N.: The why and how of nonnegative matrix factorization. In: Suykens, J., Signoretto, M., Argyriou, A. (eds.) Regularization, Optimization, Kernels, and Support Vector Machines. Machine Learning and Pattern Recognition Series. Chapman & Hall/CRC, London (2014)
21. Gillis, N., Glineur, F.: Using underapproximations for sparse nonnegative matrix factorization. Pattern Recognit. **43**(4), 1676–1687 (2010)
22. Gillis, N., Glineur, F.: Accelerated multiplicative updates and hierarchical ALS algorithms for nonnegative matrix factorization. Neural Comput. **24**(4), 1085–1105 (2012)
23. Gillis, N., Glineur, F.: On the geometric interpretation of the nonnegative rank. Linear Algebra Appl. **437**(11), 2685–2712 (2012)
24. Gillis, N., Vavasis, S.: Semidefinite programming based preconditioning for more robust near-separable nonnegative matrix factorization. SIAM J.Optim. **25**, 677–698 (2015)
25. Goemans, M.: Smallest Compact Formulation for the Permutahedron (2009). http://math.mit.edu/~goemans/PAPERS/permutahedron
26. Gouveia, J.: Personnal Comunication (2014)
27. Gouveia, J., Fawzi, H., Robinson, R.: Rational and Real Positive Srank can be Different (2014). arXiv:1404.4864
28. Gouveia, J., Parrilo, P., Thomas, R.: Lifts of convex sets and cone factorizations. Math. Oper. Res. **38**(2), 248–264 (2013)
29. Gouveia, J., Robinson, R., Thomas, R.: Worst-case Results for Positive Semidefinite Rank (2013). arXiv:1305.4600
30. Gregory, D.A., Pullman, N.J.: Semiring rank: boolean rank and nonnegative rank factorizations. J. Combin. Inform. Syst. Sci. **8**(3), 223–233 (1983)
31. Hrubeš, P.: On the nonnegative rank of distance matrices. Inf. Process. Lett. **112**(11), 457–461 (2012)
32. Janecek, A., Tan, Y.: Iterative improvement of the multiplicative update NMF algorithm using nature-inspired optimization. in Seventh International Conference on Natural Computation vol. 3 (2011), pp. 1668–1672
33. Janecek, A., Tan, Y.: Swarm intelligence for non-negative matrix factorization. Int. J. Swarm Intell. Res. **2**(4), 12–34 (2011)
34. Janecek, A., Tan, Y.: Using population based algorithms for initializing nonnegative matrix factorization. Adv. Swarm Intell. *6729*, 307–316 (2011)
35. Kaibel, V.: Extended formulations in combinatorial optimization. Optima **85**, 2–7 (2011)
36. Kaibel, V., Weltge, S.: A Short Proof that the Extension Complexity of the Correlation Polytope Grows Exponentially (2013). arXiv:1307.3543
37. Kim, J., He, Y., Park, H.: Algorithms for nonnegative matrix and tensor factorizations: a unified view based on block coordinate descent framework. J. Global Optim. **58**(2), 285–319 (2014)
38. Kim, J., Park, H.: Fast nonnegative matrix factorization: an active-set-like method and comparisons. SIAM J. Sci. Comput. **33**(6), 3261–3281 (2011)
39. Lee, D., Seung, H.: Learning the parts of objects by nonnegative matrix factorization. Nature **401**, 788–791 (1999)
40. Lee, D., Seung, H.: Algorithms for non-negative matrix factorization. In: Advances in Neural Information Processing Systems, vol. 13, pp. 556–562 (2001)

41. Lee, T., Shraibman, A.: Lower Bounds in Communication Complexity. Found. Trends Theor. Comput. Sci. **3**(4), 263–399 (2007)
42. Moitra, A.: An Almost Optimal Algorithm for Computing Nonnegative Rank. in *Proceedings of the 24th Annual ACM-SIAM Symposium on Discrete Algorithms* (SODA '13), pp. 1454–1464 (2013)
43. Oelze, M., Vandaele, A., Weltge, S.: Computing the Extension Complexities of all 4-Dimensional 0/1-polytopes (2014). arXiv:1406.4895
44. Padrol, A., Pfeifle, J.: Polygons as Slices of Higher-Dimensional Polytopes (2014). arXiv:1404.2443
45. Pirlot, M.: General local search methods. Eur. J. Oper. Res. **92**(3), 493–511 (1996)
46. Rothvoss, T.: The Matching Polytope has Exponential Extension Complexity (2013). arXiv:1311.2369
47. Shitov, Y.: Sublinear Extensions of Polygons (2014). arXiv:1412.0728
48. Shitov, Y.: An upper bound for nonnegative rank. J. Combin. Theory Ser. A **122**, 126–132 (2014)
49. Shitov, Y.: Nonnegative Rank Depends on the Field (2015). arXiv:1505.01893
50. Takahashi, N., Hibi, R.: Global convergence of modified multiplicative updates for nonnegative matrix factorization. Comput. Optim. Appl. **57**(2), 417–440 (2014)
51. Thomas, L.: Rank factorization of nonnegative matrices. SIAM Rev. **16**(3), 393–394 (1974)
52. Vandaele, A., Gillis, N., Glineur, F.: On the Linear Extension Complexity of Regular n-gons (2015). arXiv:1505.08031
53. Vavasis, S.: On the complexity of nonnegative matrix factorization. SIAM J. Optim. **20**(3), 1364–1377 (2010)
54. Watson, T.: Sampling Versus Unambiguous Nondeterminism in Communication Complexity (2014). http://www.cs.toronto.edu/~thomasw/papers/nnr
55. Yannakakis, M.: Expressing combinatorial optimization problems by linear programs. J. Comput. Syst. Sci. **43**(3), 441–466 (1991)
56. Zdunek, R.: Initialization of nonnegative matrix factorization with vertices of convex polytope. In: Artificial Intelligence and Soft Computing, vol. 7267, pp. 448–455. Lecture Notes in Computer Science (2012)