Scalable Feature Matching Across Large Data Collections

David Degras
Department of Mathematics, University of Massachusetts Boston, Boston, MA

ABSTRACT
This article is concerned with matching feature vectors in a one-to-one fashion across large collections of datasets. Formulating this task as a multidimensional assignment problem with decomposable costs (MDADC), we develop fast algorithms with time complexity roughly linear in the number of datasets and space complexity a small fraction of the data size. These remarkable properties hinge on using the squared Euclidean distance as dissimilarity function, which can reduce matching problems between pairs of datasets to n problems and enable calculating assignment costs on the fly. To our knowledge, no other method applicable to the MDADC possesses these linear scaling and low-storage properties necessary to large-scale applications. In numerical experiments, the novel algorithms outperform competing methods and show excellent computational and optimization performances. An application of feature matching to a large neuroimaging database is presented. The algorithms of this article are implemented in the R package matchFeat available at https://github.com/ddegras/matchFeat. Supplementary materials for this article are available online.

1. Introduction
Matching objects across units (e.g., subjects, digital images, or networks) based on common descriptor variables is a ubiquitous task in applied science. This problem, variously known as object matching, feature matching, data association, or assignment problem, is at the core of applications such as resource allocation (Pierskalla 1968), object tracking (Bar-Shalom, Willett, and Tian 2011; Rezatofighi et al. 2015; Wang et al. 2015), object recognition (Lowe 2001; Conte et al. 2004), navigation systems (Doherty, Foure, and Leonard 2019), image registration (Ashburner 2007; Le Moigne, Netanyahu, and Eastman 2011), optimization of communication networks (Shalom, Wong, and Zaks 2010), connectomics in neuroscience (Haxby et al. 2011; Vogelstein et al. 2015), and more.

The impetus for this work is a task in functional neuroimaging which consists in matching collections of biomarkers (more precisely, brain connectivity measures) between the subjects of a study. The matching process may serve in data exploration to provide new scientific insights and generate hypotheses. It can also be a preliminary step in a group analysis to ensure meaningful comparisons across subjects. Key aspects of the matching problem under study are: (i) the number of subjects and/or the number of biomarkers per subject may be large, posing computational challenges, (ii) for two given subjects, each biomarker of one subject must be matched to at most one biomarker of the other (one-to-one matching), and (iii) the matching must be consistent, that is, transitive across subjects (for example, denoting subjects by letters and biomarkers by numbers, if A1 is matched to B2 and B2 to C3, then A1 must be matched to C3).

This matching problem is not specific to neuroimaging and is applicable to the research fields mentioned above. It is generally relevant to multilevel or hierarchical analyses where outputs of a certain level of analysis must be matched before becoming inputs at the next level. This situation typically occurs when the outputs to be matched result from an unsupervised analysis such as clustering, segmentation, or dimension reduction.

Problem formulation. The matching problem at the core of this article is as follows. Given n sets of vectors in \( \mathbb{R}^p \) having the same cardinality, say \( \{x_{11}, \ldots, x_{1m}\}, \ldots, \{x_{n1}, \ldots, x_{nm}\} \), find permutations \( \sigma_1, \ldots, \sigma_n \) of the vector labels \( \{1, \ldots, m\} \) that minimize the sum of pairwise squared Euclidean distances within clusters \( \{x_{i\sigma_i(k)}, \ldots, x_{i\sigma_i(r)}\} \) (1 \( \leq k \leq m \)). Writing \( [r] = \{1, \ldots, r\} \) for a positive integer \( r \) and letting \( S_m \) be the set of all permutations of \( [m] \), the problem expresses as

\[
\min_{\sigma_1, \ldots, \sigma_n \in S_m} \sum_{1 \leq i < j \leq n} \sum_{k=1}^{m} \|x_{i\sigma_i(k)} - x_{j\sigma_j(k)}\|^2
\]

(1)

where \( \| \cdot \| \) denotes the Euclidean norm on \( \mathbb{R}^p \).

Problem (1) is a sum-of-squares clustering problem with the constraint that each cluster must contain exactly one vector from each set \( \{x_{i1}, \ldots, x_{im}\}, i \in [n] \). Identifying the \( n \) sets with statistical units, this constraint guarantees that the obtained clusters reflect common patterns between units, not within units.

In problem (1), all statistical units have the same number \( m \) of vectors. It is natural to also set to \( m \) the number of clusters to partition the vectors into. In practice however, statistical units...
may have unbalanced numbers of observations, say \(m_1, \ldots, m_n\). It may also be desirable to group the observations in an arbitrary number of clusters, say \(K\). Accordingly, a more general version of problem (1) would be, for each \(i \in [n]\), to assign vectors \(x_{i1}, \ldots, x_{im_i}\) to \(K\) clusters in a one-to-one fashion so as to minimize the total sum of pairwise squared Euclidean distances within clusters. Here, one-to-one means that each unit \(i\) can contribute at most one vector to any cluster. The matching problem (1) thus generalizes as

\[
\min_{s_1, \ldots, s_n} \sum_{1 \leq i < j \leq n} \sum_{k=1}^K \sum_{q \in [m_i]} \sum_{r \in [m_j]} \sum_{s_i(q) = s_j(r) = k} \|x_{ij} - x_{ij}\|^2
\]

where each \(s_i\) is a map from the set \([m_i]\) of vector labels to the set \([0, \ldots, K]\) of cluster labels where, by convention, labels of unassigned/ unmatched vectors are mapped to the cluster label 0. The map \(s_i\) is such that \(s_i(q) = s_i(r) = 0\). In other words, the restriction of \(s_i\) to \([m_i] \setminus s_i^{-1}(\{0\})\) must be an injective map. Problem (1) is recovered when \(m_1 = \cdots = m_n = K := m\), in which case \(s_i = \sigma_i^{-1}\) for all \(i \in [n]\).

For simplicity, only problem (1) is treated in this article. However, the proposed matching methods extend to the general problem (2). In complement to the model-free problem (1), a probabilistic approach to feature matching based on Gaussian mixtures is detailed in Section 2.4 and Appendix A.

Related work. Problem (1) can be viewed through the prism of combinatorial optimization problems such as the minimum weight clique partitioning problem in a complete \(n\)-partite graph, the quadratic assignment problem (Koopmans and Beckmann 1957; Cela 1998), or the multidimensional assignment problem (MAP) (e.g., Burkard, Dall’Amico, and Martello 2009). The MAP formalism is well suited to this work and is recalled hereafter:

\[
\min_{\sigma_1, \ldots, \sigma_n \in S_n} \sum_{k=1}^m c_{\sigma_1(k)\sigma_2(k)\cdots \sigma_n(k)}
\]

where \((c_{a_1a_2\cdots a_n}) \in \mathbb{R}^{m^n}\) is an \(n\)-dimensional array containing the costs of assigning the feature vectors \(x_{1a_1}, \ldots, x_{na_n}\) to the same cluster. Problem (1) is an instance of the MAP and, more precisely, it is a multidimensional assignment problem with decomposable costs (MDADC) (e.g., Bandelt, Crama, and Spieksma 1994; Bandelt, Maas, and Spieksma 2004) because its assignment costs decompose as

\[
c_{a_1a_2\cdots a_n} = \sum_{1 \leq i < j \leq n} d(x_{ia_i}, x_{ja_j})
\]

where \(d\) is a dissimilarity function. The squared Euclidean distance \(d\) used in (1) enables the development of highly efficient computational methods (see Section 2). The need for efficient computations comes from the exponential size \((m!)^n\) of the search domain \((S_n)^n\) and from the NP-hardness of (1) (when \(n \geq 3\)) as a generalization of the 3D assignment problem of Spieksma and Woeginger (1996).

The formidable literature on the MAP, which spans more than five decades and multiple mathematical fields, will not be reviewed here. The interested reader may fruitfully consult Burkard, Dall’Amico, and Martello (2009) and Pardalos and Pitsoulis (2000). In fact, given the focus of the present work on computations, a broad review of the general MAP is not necessary. Indeed, optimization methods for the MAP (e.g., Pierskalla 1968; Poore and Rijavec 1993; Robertson 2001; Karapetyan and Gutin 2011) are not computationally efficient for the special case of MDADC (and in particular (1)), especially if the number \(n\) of dimensions is large. We will therefore, only discuss the relevant MDADC literature.

Bandelt, Crama, and Spieksma (1994) provide simple "hub" and "recursive" heuristics for the MDADC (3)-(4) along with their approximation ratios (worst-case bounds on the ratio of a method’s attained objective to the optimal objective value). The hub heuristic consists in selecting one dimension \(i \in [n]\) of the MDADC as a "hub" and matching all other dimensions to this one, that is, finding for each dimension \(j \neq i\) the assignment that minimizes the total cost with respect to \(i\). The recursive heuristic starts by permuting the \(n\) dimensions of the problem and then recursively finds the best assignment for the \(i\)th permuted dimension with respect to the \((i-1)\) first permuted dimensions \((i = 2, \ldots, n)\). Bandelt, Maas, and Spieksma (2004) enhance the heuristic methods of Bandelt, Crama, and Spieksma (1994) with local neighborhood search methods that attempt to improve a solution one or two dimensions at a time. They derive lower bounds for the minimum cost assignment based on a Lagrangian relaxation of the MDADC. Collins (2012) also exploits the idea of improving solutions one dimension at a time in the general MAP (3) through a factorization technique. Kuroki and Matsui (2009) formulate (1) as the problem of finding a clique cover of an \(n\)-partite graph with minimum edge weights. They express the clique cover problem with various mathematical programs (integer linear, nonconvex quadratic, integer quadratic, and second order cone) which they tackle directly or after relaxation. They also provide approximation ratios and computational complexity bounds for their algorithms. Tauer and Nage (2013) and Natu, Date, and Nage (2020) solve Lagrangian relaxations of the integer linear program formulation of the MDADC, with an emphasis on efficient parallel computation in a Map-Reduce framework or with GPUs. They derive tight lower bounds to control the approximation error of their algorithms.

As an alternative from the multidimensional assignment perspective, problem (1) can be viewed as an instance of constrained clustering or semisupervised learning (Basu, Davidson, and Wagstaff 2009; Gancarski et al. 2020). The constraint that each unit \(i \in [n]\) contributes exactly one feature vector to each cluster can be rephrased as: two vector instances from the same unit cannot be assigned to the same cluster. This type of constraint, namely that certain pairs of instances cannot be assigned to the same cluster ("cannot link" constraint) or that certain pairs must be assigned to the same cluster ("must link" constraint), is called equivalence constraints and can be handled by constrained K-means algorithms (Wagstaff et al. 2001; Bilenko, Basu, and Mooney 2004; Pelleg and Baras 2007) or through constrained mixture models (Shental et al. 2004).

Other tasks related to problem (1) but not directly relevant are object tracking, with applications in engineering and more recently in computer vision and artificial intelligence, and image registration, which plays a key role in image processing, object
recognition, and remote sensing. The former involves a temporal dimension absent from (1) whereas the latter involves many (and often noisy) features that are not matched one-to-one. Matching problems also have a long history in statistics and have been a topic of intense scrutiny in machine learning in recent years (DeGroot and Goel 1976; Collier and Dalalyan 2016; Hsu, Shi, and Sun 2017; Pananjady, Wainwright, and Courtade 2018). However, much of the research in these fields relevant to (1) deals with the case where \( n = 2 \) and \( m \) is large (asymptotically \( m \to \infty \)) whereas we are chiefly interested in situations where \( m \) is fixed and \( n \) is large \((n \to \infty)\).

Contributions. The methods for the MDADC (3)–(4) discussed heretofore are applied in practice to problems of small size, say \( n \) in the single digits or a few tens. Theoretical considerations (Section 2) as well as numerical experiments from this paper (Section 3) and from the literature indicate that these methods cannot handle large-scale problems with \( n \) in the hundreds, thousands or more (at least, not in a reasonable time on a single computer). As a simple example, the \( \binom{n}{2}m^2 \) costs in (4) are typically calculated and stored before starting the optimization, but even this preliminary step may exceed computer memory limits for large \( n \) and/or \( m \). In response to this methodological gap, our research aims to develop fast, scalable methods for matching feature vectors in a one-to-one fashion across a large number of statistical units. The main contributions of the paper are the following.

1. We develop fast algorithms for approximately solving the matching problem (1), that is, (3)–(4) with \( d \) as the squared Euclidean distance. The two main algorithms have iteration complexity \( O(m^2n + mn^2) \) and only require a few iterations to converge in our experiments. This suggests that in practice, these algorithms may scale linearly with \( n \) although they do not have polynomial time complexity in theory (recall that problem (1) is NP-hard). In addition, they calculate assignment costs (4) on the fly and have space requirements \( O(mn + mn) \), a fraction of the data size \( mn^2 \). We also present initialization methods and a refinement method (pairwise interchange). Further, we take a probabilistic view of (1) as a constrained Gaussian mixture model and devise an efficient implementation of the Expectation-Maximization (EM) algorithm.

2. We provide a broad review of the methods applicable to (1) (integer linear programming, various relaxations, constrained clustering) which rarely appear together in an article. The novel algorithms are compared to these methods in numerical experiments and show excellent computation and optimization performances.

3. An R package matchFeat implementing all the algorithms of the article is made available at github.com/ddegras/matchFeat.

4. The matching problem (1) is applied to a large neuroimaging database to study the functional connectivity (FC) of the human brain. The data analysis recovers known FC networks but also generates new insights on the organization of networks within individuals and on their variations between individuals.

Organization of the paper. Section 2 introduces novel algorithms for problem (1). In Section 3, a numerical study compares these algorithms with benchmark methods in terms of computation and optimization performance. Section 4 presents an application of our matching approach to a large neuroimaging database (ABIDE) relating to autism spectrum disorders. Section 5 gathers concluding remarks. Appendices A and B provide additional details on the algorithms. Supplementary materials contain a proof of a theoretical result on convex relaxation and information on the neuroimaging data.

2. Novel Algorithms for Feature Matching

2.1. k-means Matching

For a given \( n \)-tuple of permutations \( \sigma = (\sigma_1, \ldots, \sigma_n) \in (S_m)^n \), let \( \bar{x}_\sigma \) be the average matrix of the permuted data with columns \( \bar{x}_{\sigma,k} := \frac{1}{n} \sum_{i=1}^{n} x_{\sigma_i(k)} \) for \( k \in [m] \). Problem (1) is equivalent to

\[
\min_{\sigma_1, \ldots, \sigma_n \in S_m} \sum_{i=1}^{n} \sum_{k=1}^{m} \|x_{\sigma_i(k)} - \bar{x}_{\sigma,k}\|^2.
\]

The following method adapts the standard k-means clustering algorithm (Lloyd 1982) to the matching problem (5).

1. Initialize \( \sigma = (\sigma_1, \ldots, \sigma_n) \) to some arbitrary value, for example \( \sigma = (Id_{[m]}, \ldots, Id_{[m]}) \). Calculate the average matrix \( \bar{x}_\sigma \) and the objective value (5).

2. Given the average matrix \( \bar{x}_\sigma \): for \( i \in [n] \), find the permutation \( \sigma_i \) that minimizes \( \sum_{k=1}^{m} \|x_{\sigma_i(k)} - \bar{x}_{\sigma,k}\|^2 \). Update the solution to \( \sigma = (\sigma_1, \ldots, \sigma_n) \).

3. Given \( \sigma \): calculate the average matrix \( \bar{x}_\sigma \) and the objective value (5). If the objective has not decreased from the previous iteration, terminate the execution and return \( \sigma \). Else go back to Step 2.

Steps 2 and 3 above are nonincreasing in the objective (5). For this reason, and due to the finiteness of the search space, the proposed approach converges in a finite number of iterations. Like the k-means, it only finds a local minimum of (5).

Concerning computations, step 3 can be performed in \( O(nmn) \) flops. Step 2, which consists of \( n \) separate optimizations, is the computational bottleneck. Observe that

\[
\sum_{k=1}^{m} \|x_{\sigma_i(k)} - \bar{x}_{\sigma,k}\|^2 = \sum_{k=1}^{m} \|x_{\sigma_i(k)}\|^2 - 2 \sum_{k=1}^{m} \langle x_{\sigma_i(k)}, \bar{x}_{\sigma,k} \rangle + \sum_{k=1}^{m} \|\bar{x}_{\sigma,k}\|^2
\]

where \( \langle \cdot, \cdot \rangle \) denotes the Euclidean scalar product. That is, the minimization of \( \sum_{k=1}^{m} \|x_{\sigma_i(k)} - \bar{x}_{\sigma,k}\|^2 \) with respect to \( \sigma_i \in S_m \) is equivalent to

\[
\max_{\sigma_i \in S_m} \sum_{k=1}^{m} \langle x_{\sigma_i(k)}, \bar{x}_{\sigma,k} \rangle.
\]
Problem (6) is an instance of the well-known linear assignment problem (LAP) (e.g., Burkard, Dell’Amico, and Martello 2009, chap. 4). After calculating the assignment matrix \( A_i = (\bar{x}_{s, k} x_{j k})_{1 \leq s, k \leq m} \), the LAP (6) can be solved for example with the Hungarian algorithm (Kuhn 1955; Munkres 1957). Efficient implementations of the Hungarian algorithm have complexity \( \mathcal{O}(m^3) \).

**Remark.** If \( p = 1 \), the matrices \( X_i \) are row vectors and the \( x_{ik} \) are scalars. In this case, solving the LAP (6) is trivial. Indeed for each \( i \in [n] \), the sum \( \sum_{k=1}^{m} x_{ik} \bar{x}_{s, k} \) is maximized when the \( x_{ik} \) and \( \bar{x}_{s, k} \) are matched by rank. More precisely, take any \( s_i \in S_m \) such that \( x_{is_i(1)} \leq \cdots \leq x_{is_i(m)} \) and \( s \in S_m \) such that \( \bar{x}_{s, s(1)} \leq \cdots \leq \bar{x}_{s, s(m)} \). Then \( s_i = s \circ s^{-1} \) maximizes the sum. In other words, the optimal permutations \( s_i \) are simply obtained by sorting the components of the \( X_i \) and \( \bar{x}_{s, k} \) (computational complexity \( \mathcal{O}(mn \log m) \)).

The \( k \)-means matching algorithm is summarized hereafter. The objective value in (5) is denoted by \( F(\sigma) \).

**Algorithm 1** \( k \)-Means Matching

Require: \( X_1, \ldots, X_n \in \mathbb{R}^{F \times m}, \sigma = (\sigma_1, \ldots, \sigma_n) \in (S_m)^n \)

1. \( \bar{x}_{s, k} = ((1/n) \sum_{i=1}^{n} x_{i j k})_{j \in [m]} \), \( F_{\text{new}} \leftarrow F(\sigma) \)

2. repeat

3. \( F_{\text{old}} \leftarrow F_{\text{new}} \)

4. for \( i = 1, \ldots, n \) do

5. Solve the LAP (6) and call \( \sigma_i^+ \) a solution.

6. \( \sigma_i \leftarrow \sigma_i^+ \)

7. end for

8. \( \sigma \leftarrow (\sigma_1, \ldots, \sigma_n) \)

9. \( \bar{x}_{s, k} = ((1/n) \sum_{i=1}^{n} x_{i j k})_{j \in [m]} \), \( F_{\text{new}} \leftarrow F(\sigma) \)

10. until \( F_{\text{new}} \geq F_{\text{old}} \)

**2.2. Block Coordinate Ascent Method**

For convenience problem (1) is rewritten here using permutation matrices \( P_1, \ldots, P_n \) instead of permutation functions \( \sigma_1, \ldots, \sigma_n \). Each \( P_i \) (\( i \in [n] \)) is a square matrix with entries in \( \{0, 1\} \) such that each row and each column contains the value 1 exactly once. Let \( \Pi_n \) be the set of all \( m \times m \) permutation matrices. Problem (1) expresses (up to a factor 1/2) as the binary quadratic assignment problem

\[
\min_{P_1, \ldots, P_n \in \Pi_n} \sum_{i=1}^{n} \sum_{j=1}^{n} \|X_i P_i - X_j P_j\|_F^2
\]

where \( \| \cdot \|_F \) denotes the Frobenius norm \( \|X\|_F = \langle X, X \rangle_F^{1/2} = (\text{tr}(X^T X))^{1/2} \) with \( \text{tr}(\cdot) \) the trace operator. By expanding the squared Frobenius norm in the objective and noting that column permutations do not change the Frobenius norm of a matrix, we have

\[
\sum_{i=1}^{n} \sum_{j=1}^{n} \|X_i P_i - X_j P_j\|_F^2 = \sum_{i=1}^{n} \sum_{j=1}^{n} (\|X_j P_j\|_F^2 + \|X_j P_j\|_F^2 - 2\langle X_i P_i, X_j P_j \rangle_F)
\]

Discarding terms that do not depend on \( P_1, \ldots, P_n \), problem (7) is equivalent to

\[
\max_{P_1, \ldots, P_n \in \Pi_n} \left\| \sum_{i=1}^{n} X_i P_i \right\|_F^2
\]

The maximization problem (8) can be handled one matrix \( P_i \) at a time (\( i \in [n] \)), that is, by block coordinate ascent (BCA, e.g., Wright 2015). Given a current solution \( (\hat{P}_1, \ldots, \hat{P}_n) \) and an index \( i \), all matrices \( \hat{P}_j, j \neq i \), are fixed and the task at hand is

\[
\max_{P_i \in \Pi_m} \left\{ X_j P_j + \sum_{j \in [n] \setminus \{i\}} X_j \hat{P}_j \right\}_F^2
\]

As mentioned in Section 2.1, (9) can be efficiently solved with the Hungarian algorithm. The permutation matrix \( \hat{P}_i \) is then updated to a solution of (9). This operation is repeated with the index \( i \) sweeping through the set \( [n] \) until no further increase in the objective (8) has been achieved in a full sweep. Given that each update of a \( \hat{P}_i \) is nondecreasing in the objective (8) and that the search domain \( (\Pi_m)^n \) is finite, the algorithm is guaranteed to converge in a finite number of steps. Popular methods for sweeping through \( [n] \) include the cyclical order, random sampling, random permutation of \( [n] \), and greedy selection.

The BCA algorithm is summarized hereafter. The objective in (8) is denoted by \( F \). For simplicity the sweeping order is taken to be cyclical but any sweeping method can be used.

**Algorithm 2** Block Coordinate Ascent

Require: \( X_1, \ldots, X_n \in \mathbb{R}^{F \times m}, P_1, \ldots, P_n \in \Pi_m \)

1. \( S \leftarrow \sum_{i=1}^{m} X_i P_i, F_{\text{new}} \leftarrow \|S\|_F^2 \)

2. repeat

3. \( F_{\text{old}} \leftarrow F_{\text{new}} \)

4. for \( i = 1, \ldots, n \) do

5. \( S_i \leftarrow S - X_i P_i \)

6. Solve the LAP \( \max_{P_i \in \Pi_m} \{P_i, X_i S_i\}_F \) and call \( P_i^+ \) a solution.

7. \( P_i \leftarrow P_i^+, S \leftarrow S_i + X_i P_i \)

8. end for

9. \( F_{\text{new}} \leftarrow \|S\|_F^2 \)

10. until \( F_{\text{new}} \leq F_{\text{old}} \)

Algorithm 2 can be viewed as a special case of the local search algorithm LS1 of Bandelt, Maas, and Spieksma (2004). The LS1 algorithm is more general in that it uses an arbitrary dissimilarity function \( d \) in the MDADC (3)–(4). The computational price to pay for this generality is that for each block update (\( i \in [n] \))
the assignment matrix $A_i = \sum_{j=1}^{n} d(x_{\sigma_i(j)}, x_{\sigma_i(j)})$ must be calculated from scratch in $O(m^2 np)$ flops. Hence, the LS1 method has iteration complexity $O(m^2 np)$ (one iteration meaning one sweep through $\{i\}$) which may be prohibitive for large $n$. In comparison, the squared Euclidean distance $d = \| \cdot \|^2$ employed in the BCA method enables efficient computation of $A_i$ in $O(m^2 p)$ complexity by keeping track of the running sum $\sum_{n} X_i P_i$ with rank-1 updates. Accordingly, the BCA method has iteration complexity $O(m^3 2n + m^2 np)$ linear in $n$.

Algorithm 2 is superior to the $k$-means matching Algorithm 1 in the sense that it can improve a solution produced by Algorithm 1 whereas the converse is not true. This is because in the assignment step of Algorithm 1, the search for the best permutation $\sigma_i (i \in [n])$ in (5) proceeds with the cluster centers $\bar{x}_{\sigma,k} (\sigma \in [m])$ fixed and thus does not account for the fact that both the $x_{\sigma_i(k)}$ and the $\bar{x}_{\sigma,k}$ depend on $\sigma$. Accordingly, this assignment step minimizes a proxy of the objective (5).

In contrast, the assignment step of Algorithm 2 minimizes (5) itself (or equivalently maximizes (8)) for each $\sigma_i$ or $P_i$. Related to this, Appendix B shows that Algorithm 1 solves a relaxed version of (8). Another factor contributing to the superiority of Algorithm 2 is that this algorithm assigns the permutations $\sigma_i$ in a sequential fashion, so that later assignments are informed by earlier ones, whereas Algorithm 1 performs these assignments independently.

### 2.3. Pairwise Interchange Heuristic

The BCA algorithm of Section 2.2 attempts to improve an existing solution to (1) one permutation $\sigma_i$ at a time: at each step, it changes all assignments $\sigma^i = (\sigma_1(i), \ldots, \sigma_n(l)) (l \in [m])$ in a single dimension $i \in [n]$ Karapetyan and Gutin (2011) call this approach a $dimensionwise$ heuristic. Another strategy called the interchange or $k$-exchange heuristic is to change a few assignments (typically, $k \in (2, 3)$) in all dimensions by element swaps (e.g., Balas and Saltzman 1991; Robertson 2001; Oliveira and Pardalos 2004). Here we consider the 2-exchange algorithm (Algorithm 3.4) of Robertson (2001) for the general MAP (3) and adapt it to (1). In this algorithm, given two assignments, the search for the best interchange is done exhaustively. This involves accessing as many as $2^n - 1$ candidate assignments for element swaps and comparing their costs, which is reasonable in the general MAP provided that costs are precalculated and that $n$ is small. However, for moderate to large $n$, and in the context of (1) where assignment costs are not precalculated, the calculation of $2^n - 1$ interchange assignment costs for each of $\binom{n}{2}$ candidate pairs of assignments (per algorithm iteration) is intractable. We will show that in (1), the pairwise interchange heuristic can be efficiently solved as a binary quadratic program.

Given a solution $\sigma = (\sigma_1, \ldots, \sigma_n)$ to (1) and two associated assignments $\sigma^i$ and $\sigma^{i'}$ ($1 \leq q < r \leq m$), the basic problem of pairwise interchange is to improve the objective in (1) by interchanging elements between these assignments, that is, by swapping the values of $\sigma_i(q)$ and $\sigma_i(r)$ for one or more indices $i \in [n]$.

Formally, the problem is under the constraints

$$
\begin{aligned}
\min_{\sigma^*_1, \ldots, \sigma^*_n \in S_m} & \sum_{1 \leq i < j \leq n} \sum_{k=1}^{m} \|X_{\sigma^*_i(k)} - X_{\sigma^*_j(k)}\|^2 \\
\text{subject to} & \sigma_i^{i'}(k) = \sigma_i(k), \ k \in [m] \setminus \{q,r\} \\
& (\sigma_i^{i'}(q), \sigma_i^{i'}(r)) \in \{(\sigma_i(q), \sigma_i(r)), (\sigma_i(r), \sigma_i(q))\}, \ i \in [n].
\end{aligned}
$$

(10a)

To fix ideas, assume without loss of generality that $(q, r) = (1, 2)$ and $\sigma_1 = \text{Id}_{[m]}$ for $i \in [n]$. Problem (10) becomes

$$
\begin{aligned}
\min_{\sigma^*_1, \ldots, \sigma^*_n \in S_2} & \sum_{1 \leq i < j \leq n} \|X_{\sigma^*_i(1)} - X_{\sigma^*_j(1)}\|^2 + \sum_{1 \leq i \leq n} \|X_{\sigma^*_i(2)} - X_{\sigma^*_j(2)}\|^2.
\end{aligned}
$$

(10b)

As in the previous sections, the problem can be transformed to

$$
\begin{aligned}
\max_{\sigma^*_1, \ldots, \sigma^*_n \in S_2} & \sum_{i=1}^{n} \|X_{\sigma^*_i(1)}\|^2 + \sum_{i=1}^{n} \|X_{\sigma^*_i(2)}\|^2.
\end{aligned}
$$

Replacing the permutations $\sigma^*_i \in S_2$ by binary variables $c_i$, the problem becomes

$$
\begin{aligned}
\max_{c_1, \ldots, c_n \in [0, 1]} & \sum_{i=1}^{n} (c_i x_{i1} + (1-c_i) x_{i2})^2 + \sum_{i=1}^{n} ((1-c_i) x_{i1} + c_i x_{i2})^2 \quad \text{and, after simple manipulations,}
\end{aligned}
$$

$$
\begin{aligned}
\max_{c_1, \ldots, c_n \in [0, 1]} & \sum_{i=1}^{n} c_i c_j (d_i, d_j) - n \sum_{i=1}^{n} c_i (d_i, a)
\end{aligned}
$$

(12)

where $d_i = x_{i1} - x_{i2}$ and $a = (1/n) \sum_{i=1}^{n} d_i$. This is an unconstrained binary quadratic program (UBQP) of size $n$ that can be solved with standard mathematical software (e.g., Cplex, Gurobi, Mosek). Refer to Kochenberger et al. (2014) for details on UBQPs.

Having reduced the basic pairwise interchange problem (10) to the UBQP (12), we now embed it in Algorithm 3.4 of Robertson (2001) which combines randomization and greedy selection of interchange pairs. Hereafter $F(\sigma)$ denotes the objective value in (1) and $\sigma = (\sigma_1, \ldots, \sigma_n) \in (S_m)^n$ is identified with the assignments $[\sigma^1, \ldots, \sigma^m]$, where $\sigma^1 = (\sigma_1(i), \ldots, \sigma_n(l))$. The notation $\text{diag}(\cdot)$ is used for diagonal matrices.

### 2.4. Gaussian Mixture Approach

Problem (1) has a probabilistic interpretation in terms of mixture models. Let $y_1, \ldots, y_m$ be random vectors in $\mathbb{R}^p$ with respective probability distributions $P_1, \ldots, P_m$. Assume that these vectors are observed after their labels have been shuffled at random. The random permutation of labels represents the uncertainty about the correspondence between observations, say $x_1, \ldots, x_m$, and their underlying distributions $P_1, \ldots, P_m$. For mathematical convenience, $y_1, \ldots, y_m$ are assumed independent and each $P_k (k \in [m])$ is taken as a multivariate normal distribution $N(\mu_k, \Sigma_k)$. The data generation process expresses as

$$
\begin{aligned}
y_k \sim N(\mu_k, \Sigma_k) \quad (k \in [m]),
\end{aligned}
$$

(13)

$s$ has a uniform distribution over $S_m$, (y_1, \ldots, y_m)$ are mutually independent and independent of $s$,

$$
\begin{aligned}
(x_1, \ldots, x_m) = (y_{s(1)}, \ldots, y_{s(m)}).
\end{aligned}
$$

Model (13) can be formulated as a Gaussian mixture model (GMM). Indeed, after concatenating observations, the vector
Algorithm 3 Pairwise Interchange with Greedy Selection

Require: $X_1, \ldots, X_n \in \mathbb{R}^{p \times m}$, $\sigma \equiv \{\sigma^1, \ldots, \sigma^m\}$
1: $C \leftarrow \sigma$ [candidate set of assignments for interchange]
2: while $C \neq \emptyset$ do
3: $F_{best} \leftarrow F(\sigma)$
4: $\sigma^+ \leftarrow \emptyset$, $\tau^+ \leftarrow \emptyset$
5: Select $\sigma^+ \in C$
6: for $\sigma^+ \in C \setminus \{\sigma^i\}$ do
7: $d_i \leftarrow x_{i\sigma^i}(i) - x_{\sigma^+}(i)$ ($i \in [n]$), $\tilde{d} \leftarrow \frac{1}{n} \sum_{i=1}^{n} d_i$
8: $Q \leftarrow ((d_i, d_i))_{1 \leq i, j \leq m} - \text{diag}(n(d_1, d_1), \ldots, n(d_1, d_1))$
9: Solve the UBQP (12) with quadratic matrix $Q$ and call $(c_1, \ldots, c_n)$ a solution.
10: $\tilde{\sigma}^q(i) \leftarrow c_j \sigma^q(i) + (1 - c_j) \sigma^r(i)$ ($i \in [n]$)
11: $\tilde{\sigma}^r(i) \leftarrow c_j \sigma^r(i) + (1 - c_j) \sigma^q(i)$ ($i \in [n]$)
12: $\tilde{F} \leftarrow F(\sigma \setminus \{\sigma^q, \sigma^r\} \cup \{\tilde{\sigma}^q, \tilde{\sigma}^r\})$
13: if $\tilde{F} < F_{best}$ then
14: $(\sigma^+, \tau^+) \leftarrow (\tilde{\sigma}^q, \tilde{\sigma}^r)$ [candidate new pair of assignments]
15: $(\sigma^-, \tau^-) \leftarrow (\sigma^q, \sigma^r)$ [candidate old pair of assignments]
16: $F_{best} \leftarrow \tilde{F}$
17: end if
18: end for
19: if $\sigma^+ \neq \emptyset$ then
20: $\sigma \leftarrow \sigma \setminus \{\sigma^-, \tau^+\} \cup \{\sigma^+, \tau^+\}$ [perform interchange]
21: $C \leftarrow \sigma$ [reset candidate set to all assignments]
22: else
23: $C \leftarrow C \setminus \{\sigma^q\}$ [remove assignment from candidate set]
24: end if
25: end while

$x = \text{vec}(x_1, \ldots, x_m)$ follows a mixture of $m$ distributions $N(\mu_{\sigma}, \Sigma_{\sigma})$ in $\mathbb{R}^{mp}$ with equal mixture weights $1/m!$, where $\mu_{\sigma} = \text{vec}(\mu_{\sigma^1}, \ldots, \mu_{\sigma^m})$ and $\Sigma_{\sigma} = \text{diag}(\Sigma_{\sigma^1}, \ldots, \Sigma_{\sigma^m})$ (block-diagonal matrix) for $\sigma \in S_m$; see (Kim 2008, chap. 4) for details. Hence, the theory and methods of GMMs apply to (13), in particular the consistency and asymptotic normality of maximum likelihood estimators (McLachlan and Peel 2000, chap. 2). For computations, the Expectation Maximization (EM) algorithm is arguably the preferred approach to maximum likelihood estimation in GMMs (e.g., McLachlan and Peel 2000). Although the standard EM algorithm for GMMs can in principle be applied to the mixture formulation of (13), the number $m$ of mixture components and the dimension $mp$ of the concatenated data render computations intractable unless $m$ is very small. In Appendix A, we propose a tractable EM for model (13).

Remark. In model (13), the cluster centers $\{\hat{x}_{\sigma^1}, \ldots, \hat{x}_{\sigma^m}\}$ associated to a global solution $\hat{\sigma} = (\hat{\sigma}_1, \ldots, \hat{\sigma}_n)$ of (1) are in general not consistent for $\{\mu_1, \ldots, \mu_n\}$ as $n \to \infty$. For example, if $p = 1$, $m = 2$, and $\mu_1 < \mu_2$, then $\hat{\mu}_1 = \frac{1}{n} \sum_{i=1}^{n} \min(x_{1i}, x_{2i})$ and $\hat{\mu}_2 = \frac{1}{n} \sum_{i=1}^{n} \max(x_{1i}, x_{2i})$. Accordingly $E(\hat{\mu}_1) = E(\min(x_{1i}, x_{2i})) < \mu_1$ and $E(\hat{\mu}_2) = E(\max(x_{1i}, x_{2i})) > \mu_2$, meaning that both estimators are biased and inconsistent.

Remark. The permutation in (13) can be formulated as equivalence constraints (see Shental et al. 2004, and Section 1). However, this general formulation is unlikely to lead to faster or better optimization, just as the constrained $k$-means approach of Wagstaff et al. (2001), which also handles equivalence constraints, does not improve upon the specialized $k$-means Algorithm 1 for problem (1) (see Section 3).

2.5. Algorithm Initialization

The matching methods developed in the previous sections are local search procedures. As such, the quality of their solutions largely depends on their starting points. Several strategies for finding good starting points are presented hereafter.

Fixed initialization. The permutation functions $\sigma_1, \ldots, \sigma_n$ or matrices $P_1, \ldots, P_n$ can be initialized to fixed values, for example the identity function $I_{[m]}$ or matrix $I_m$. While convenient, this initialization is somewhat arbitrary unless the data are approximately matched. It is not suitable on its own but can serve as a baseline.

Random initialization. Using multiple random starting points $\sigma \in (S_m)^n$ or $P \in (\Pi_m)^n$ often yields at least one nearly optimal solution. This strategy is particularly suitable when the computational cost of optimization is cheap as with Algorithms 1–2.

Template matching. Given data matrices $X_1, \ldots, X_m \in \mathbb{R}^{p \times m}$ and a template matrix $T \in \mathbb{R}^{p \times m}$, solve the matching problem

$$\min_{P_1, \ldots, P_n \in \Pi_m} \sum_{i=1}^{n} \|X_i P_i - T\|^2_F. \quad (14)$$

The expediency of template matching comes from the fact that it reduces (2) related matching tasks between pairs of data matrices in (1) to $n$ separate matching tasks between the data and the template. A central question is: which template to use? Bandelt, Crama, and Spieksma (1994) propose to either take a single data matrix as template (single hub heuristic), for example, $T = X_1$, or to examine all data matrices in turn: $T \in \{X_1, \ldots, X_n\}$, and retain the assignment $P(T) = (P_1(T), \ldots, P_n(T))$ that yields the lowest value of (1) (multiple hub heuristic). More generally, the template need not be a data point; it could for example be an estimate of cluster centers based on previous observations.

Recursive heuristics. The recursive heuristics of Bandelt, Crama, and Spieksma (1994) (see Section 1) are easily applicable to problem (1). Their algorithm RECUR1 for example, which is related to the BCA Algorithm 2, is implemented as follows. The first permutation matrix $P_1$ can be selected arbitrarily, say $P_1 = I_m$. Then for $i = 1, \ldots, n - 1$, the LAP (9) is changed to

$$\max_{P_{i+1} \in \Pi_m} \left\{ P_{i+1}, X_{i+1}^T \sum_{j=1}^{i} X_j P_j \right\}_F. \quad (15)$$

3. Numerical Study

This section presents experiments that assess the matching algorithms of Section 2 and benchmarks from the literature in terms of numerical and computational performance. Three performance measures are reported: attained objective in problem (1), Rand index (Rand 1971) of agreement between matchings and data labels, and computation time.
**Simulation setup.** The simulations are based on handwritten digits data available on the UCI machine learning repository (archive.ics.uci.edu). Unlike classification problems, the task at hand is to match collections of digits without using label information. The data are normalized bitmaps of handwritten digits. The data are normalized bitmaps of handwritten digits. After downsampling, images of dimensions $8 \times 8$ are obtained with integer elements in $[0, \ldots, 16]$. The training data used for the simulations contain 3823 images contributed by 30 people, with about 380 examples for each digit $0, \ldots, 9$. A principal component analysis (PCA) is carried out separately for each of the $m = 10$ digit classes (after vectorizing the $8 \times 8$ input matrices) and the 25 first PCs are retained for each class, which represents at least 95% of the class variance. Artificial data are then generated according to the model $x_{ik} = \mu_k + \sum_{r=1}^{25} \xi_{ikr} \phi_k + \varepsilon_{ikr}$ for $i \in [n]$ and $k \in [m]$, where $\mu_k$ is the class mean, the $\phi_k$ are PC vectors of length $p = 64$ and the $\xi_{ikr}$ are independent normal random variables with mean zero and standard deviation given by the PCA. A small amount of Gaussian white noise $\varepsilon$ with standard deviation 2.5 is added to the simulated data, which corresponds to 95% of the standard deviation of the original data. The simulation results stay qualitatively similar when $\sigma$ varies in $[0, 12.5]$.) The number $n$ of statistical units varies in $\{5, 10, 20, 30, 40, 50, 75, 100, 200, 500, 1000\}$. For each value of $n$, the simulation is replicated 100 times. The simulations are run in the R programming environment. Code for the simulations and the R package matchFeat implementing the methods of this article are available at github.com/ddegras/matchFeat.

**Matching methods.** The methods of Section 2 are combined in three steps: initialization, main algorithm, and optional post-processing. Four initializations are considered: identity matrix (ID), 100 random starting points (R100), multiple hub heuristic (HUB), and recursive heuristic (REC). A fifth initialization clustering data vectors by their digit labels (LBL) is also examined as a benchmark. This initialization is infeasible in practice; it may also not minimize (1) although it is often nearly optimal. The main algorithms are $k$-means matching (KMM) and block coordinate ascent (BCA). The pairwise interchange algorithm (2X) and EM algorithm for constrained Gaussian mixture (EM) are used for post-processing only as they performed poorly on their own (i.e., with any of the proposed initializations) in preliminary experiments. The simulations also comprise matching methods representative of the literature:

- **Integer linear program (ILP).** The standard ILP formulation of the MDADC (3)–(4) (e.g., Kuroki and Matsui 2009) involves $\binom{n}{2} m^2$ binary variables (the number of edges in a complete $n$-partite graph with $m$ nodes in each subgraph), $n(n-1)m$ equality constraints and $\binom{n}{2} m^3$ inequality constraints (so-called triangle or clique constraints).

- **ILP relaxation and integer quadratic program (IQP).** Two methods of Kuroki and Matsui (2009) are considered: the first consists in dropping the triangle constraints, solving $\binom{n}{2}$ separate assignment problems, and recovering a proper solution with multiple-hub heuristics. The second expresses the triangle constraints with reference to one of the $n$ subgraphs—the hub—and formulates the objective function only in terms of the edges associated with the hub. This reduces the number of optimization variables to $(n-1)m^2$ but transforms the linear program into a quadratic one.

- **Constrained $k$-means.** The COP-KMEANS (Wagstaff et al. 2001), MPCK-MEANS (Bilenko, Basu, and Mooney 2004), LCVQE (Pelleg and Baras 2007), and CCLS (Hiep, Duc, and Trung 2016) algorithms all handle equivalence constraints and can thus be applied to (1). They are implemented in the R package conclust of the last authors. COP-KMEANS and CCLS treat equivalence constraints as hard constraints and thus exactly solve (1). MPCK-MEANS and LCVQE handle equivalence constraints as soft constraints (MPCK-MEANS also incorporates metric learning) and thus approximately solve (1).

Going forward, these methods will be referred to as ILP, KUR-ILP, KUR-IQP, COP-KM, MPC-KM, LCVQE, and CCLS. Lagrangian heuristics (e.g., Tauer and Nagi 2013; Natu, Date, and Nagi 2020) are not included in the simulations because their efficient implementation requires computer clusters and/or specialized computing architecture, whereas the focus of this article is on methods executable on a single machine.

**Remark.** Initial attempts were made to obtain lower bounds on the global minimum in (1) using a relaxation method of Bandelt, Maas, and Spieksma (2004). However, the resulting bounds are far too small, a fact already noted by these authors in the case of non-Euclidean distances $d$ (recall that in (1), $d$ is the squared Euclidean distance).

**Results.** Optimization accuracy. To facilitate comparisons, we discuss the relative error of each method averaged across 100 replications for each $n$. The relative error of a method is defined as the ratio of its attained objective value in (1) by the minimum objective value across all methods minus 1. Full results are available in Table 1. Hereafter and in the table, methods are listed by order of best performance.

R100-BCA is the best method for each $n$, attaining the best objective value in virtually every replication. For small values $n \in \{5, 10\}$, ILP and KUR-IQP also achieve best performance. The next best methods are LBL-BCA-2X, HUB-BCA-2X, LBL-BCA, and HUB-BCA, with a relative error decreasing from order $10^{-3}$ for $n = 5$ to order $10^{-4}$ or $10^{-5}$ for $n = 100$. Recall that the LBL initialization is an oracle of sorts since data labels are typically not available in matching problems. The methods based on $k$-means matching (KMM) yield slightly higher yet comparable relative error that goes roughly from order $10^{-2}$ for $n = 5$ to the range $(10^{-4}, 10^{-6})$ for $n = 100$. As can be expected, the ID and REC initializations yield slightly worse performance whereas R100 provides the best results. BCA is less sensitive to the initialization methods than KMM. EM, which is initialized with ID-BCA, gives reasonable results for $n \leq 50$ (relative error of order $10^{-3}$) although it does not improve upon BCA. For $n > 50$ however, its performance with respect to (1) severely deteriorates and its relative error climbs to about 0.4. Among the competitor methods, KUR-ILP has the best performance, with a relative error of order $10^{-2}$ across values of $n$. COP-KM and MPC-KM have relative errors that decrease from order $10^{-1}$ for small $n$ to order $10^{-2}$ for $n = 100$. LCVQE has a slowly decreasing relative error that goes from 0.3 for $n = 5$ to 0.2 for $n = 100$. CCLS sees its relative error increase from order $10^{-2}$ for small $n$ to 0.4 for $n = 100$. Go
Table 1. Optimization performance.

| Method          | n = 5 | n = 10 | n = 20 | n = 30 | n = 40 | n = 50 | n = 75 | n = 100 | n = 200 | n = 500 | n = 1000 |
|-----------------|-------|--------|--------|--------|--------|--------|--------|--------|--------|--------|---------|
| 2X              | 1E-2  | 1E-2   | 1E-2   | 1E-2   | 1E-2   | 1E-2   | 1E-2   | 1E-2   | 1E-2   | 1E-2   | 1E-2    |
| CCLS            | 1E-2  | 1E-2   | 1E-2   | 1E-2   | 1E-2   | 1E-2   | 1E-2   | 1E-2   | 1E-2   | 1E-2   | 1E-2    |
| KO                   | 1E-2  | 1E-2   | 1E-2   | 1E-2   | 1E-2   | 1E-2   | 1E-2   | 1E-2   | 1E-2   | 1E-2   | 1E-2    |
| HUB-BCA         | 1E-3  | 1E-3   | 1E-3   | 1E-3   | 1E-3   | 1E-3   | 1E-3   | 1E-3   | 1E-3   | 1E-3   | 1E-3    |
| ID-BCA           | 1E-3  | 1E-3   | 1E-3   | 1E-3   | 1E-3   | 1E-3   | 1E-3   | 1E-3   | 1E-3   | 1E-3   | 1E-3    |
| LBL-BCA          | 1E-3  | 1E-3   | 1E-3   | 1E-3   | 1E-3   | 1E-3   | 1E-3   | 1E-3   | 1E-3   | 1E-3   | 1E-3    |
| LBL-BCA-2X      | 1E-3  | 1E-3   | 1E-3   | 1E-3   | 1E-3   | 1E-3   | 1E-3   | 1E-3   | 1E-3   | 1E-3   | 1E-3    |
| R100-BKA        | 1E-1  | 1E-1   | 1E-1   | 1E-1   | 1E-1   | 1E-1   | 1E-1   | 1E-1   | 1E-1   | 1E-1   | 1E-1    |
| R100-BKA-2X     | 1E-1  | 1E-1   | 1E-1   | 1E-1   | 1E-1   | 1E-1   | 1E-1   | 1E-1   | 1E-1   | 1E-1   | 1E-1    |
| REC-BCA          | 1E-3  | 1E-3   | 1E-3   | 1E-3   | 1E-3   | 1E-3   | 1E-3   | 1E-3   | 1E-3   | 1E-3   | 1E-3    |
| REC-BCA-2X      | 1E-3  | 1E-3   | 1E-3   | 1E-3   | 1E-3   | 1E-3   | 1E-3   | 1E-3   | 1E-3   | 1E-3   | 1E-3    |
| HUB-BCA-2X      | 1E-3  | 1E-3   | 1E-3   | 1E-3   | 1E-3   | 1E-3   | 1E-3   | 1E-3   | 1E-3   | 1E-3   | 1E-3    |
| ID-BCA-2X        | 1E-3  | 1E-3   | 1E-3   | 1E-3   | 1E-3   | 1E-3   | 1E-3   | 1E-3   | 1E-3   | 1E-3   | 1E-3    |
| LBL-BCA-2X       | 1E-3  | 1E-3   | 1E-3   | 1E-3   | 1E-3   | 1E-3   | 1E-3   | 1E-3   | 1E-3   | 1E-3   | 1E-3    |
| LBL-BCA-2X      | 1E-3  | 1E-3   | 1E-3   | 1E-3   | 1E-3   | 1E-3   | 1E-3   | 1E-3   | 1E-3   | 1E-3   | 1E-3    |
| ID-BCA-2X       | 1E-3  | 1E-3   | 1E-3   | 1E-3   | 1E-3   | 1E-3   | 1E-3   | 1E-3   | 1E-3   | 1E-3   | 1E-3    |
| LBL-BCA-2X     | 1E-3  | 1E-3   | 1E-3   | 1E-3   | 1E-3   | 1E-3   | 1E-3   | 1E-3   | 1E-3   | 1E-3   | 1E-3    |
| R100-BKA        | 1E-1  | 1E-1   | 1E-1   | 1E-1   | 1E-1   | 1E-1   | 1E-1   | 1E-1   | 1E-1   | 1E-1   | 1E-1    |
| R100-BKA-2X    | 1E-1  | 1E-1   | 1E-1   | 1E-1   | 1E-1   | 1E-1   | 1E-1   | 1E-1   | 1E-1   | 1E-1   | 1E-1    |

**Rand index.** The Rand index is a measure of agreement between two partitions of a set; it is suitable for matching problems which produce clusters and not individual label predictions. Here the data partition produced by a matching method is compared to the partition induced by the data classes, that is, their underlying digits in [0, ..., 9]. While the goal of matching is to produce homogeneous data clusters and not to maximize agreement between the produced clusters and some underlying class/label-induced clusters, these two goals are aligned in the simulations because data vectors generated by a same digit class tend to be much closer to each other than to vectors generated by other digit classes.

Given a set D of size n and two partitions X and Y of D into clusters, the RI is defined as the ratio of \((a + b)/\binom{n}{2}\), where a is the number of pairs of elements in D that are in a same cluster both in X and Y, and b is the number of pairs of elements in D that are in different clusters both in X and Y. This can be interpreted as the fraction of correct decisions to assign two elements of D either to the same cluster or to different ones.

The RI of each method (averaged across 100 replications) is displayed in Figure 1 as a function of n. Values closer to 1 indicate better agreement between matching outputs and data labels (digits). For BCA and KMM, the RI starts from a baseline in the range [0.92, 0.96], reaches 0.99 around n = 100, and then stays at this level for n > 100. The REC initialization has a RI that increases from 0.94 for n = 5 to 0.98 for n = 100. For COP-KM, MPP-KM, LCVQ, KUR-ILP, and HUB, the RI slowly increases from about 0.9 to 0.95 with n. The initializations R100 and ID are independent of the data labels, which are randomly shuffled. They amount to random guessing and their baseline RI = 0.82 matches the theoretical expectation \((1 - (2m - 2)/m^2)\). The RI of EM and CCLS rapidly decreases at or below random guessing levels, in accord with the modest performance of these methods in the optimization (1).

**Running time.** The running times of the algorithms are displayed in Figure 2. During the simulations, algorithms were given 300 sec to complete execution, after which they were interrupted. Accordingly any value 300 on the figure (often largely) underestimates the actual computation time. The algorithms can be divided in two groups: those who can solve (1) for n = 1000 in 100 sec or far less, and those that time out (execution time over 300 sec) for n ≤ 500 or far less. They are described below by order of decreasing speed.

BCA and KMM are the fastest methods with running times of order 10⁻³ to 10⁻¹ seconds across values of n. For n = 1000, they are one order of magnitude faster than the next best method (LCVQ). The HUB and REC initializations, although slower than arbitrary starting points like identity or random permutations, enable overall faster computations because good starting points reduce the number of iterations required for the main algorithm to converge. Completion of (100 runs) of BCA or KMM based on the R100 initialization takes between 200 and 250 times the execution of a single run based on HUB or REC (instead of roughly 100). This is because the latter heuristics find good starting points whereas some (or many) of the 100 random starting points will be bad and require many iterations for the main algorithm to converge. KUR-ILP enjoys the same speed as the BCA and KMM for small n but its running time appears to scale polynomially with n. LCVQ appears to scale linearly with n but with a much larger multiplicative constant than BCA.

**NOTE:** The relative error of each method is displayed (averaged across 100 replications, standard deviation in parentheses). For each n and each group of methods (from top to bottom: benchmarks, 2X, and EM; BCA; KMM initialization), the best performance is in bold. Missing values are due to execution timeout (> 300 sec).
Figure 1. Rand index versus sample size $n$ (average across 100 replications).

Figure 2. Running time versus sample size $n$ (average across 100 replications).
and KMM. Its running time is of order $10^{-2}$ s for $n = 5$ and 1s for $n = 75$. The running time of CCLS grows steeply with $n$ and exceeds the 300 sec limit for $n \geq 500$. MPC-KM, COP-KM and EM are very slow, at least in their R implementation, and they time out (i.e., their execution times exceed 300 sec) for $n \geq 200$. Their computational load seems to grow exponentially with $n$. In the case of the EM, the computational bottleneck is the evaluation of matrix permanents. ILP, KUR-IQP and 2X are by far the slowest methods in the simulations. The first two stall and time out as soon as $n$ exceeds a few units, although they produce good results when $n \leq 5$. The computational load of 2X (following BCA or KMM) scales exponentially with $n$ (average computation time 110s for $n = 30$).

**Summary of simulations.**

- BCA is the fastest and most accurate of all studied methods. It provides excellent accuracy when initialized with REC or HUB. For best accuracy, the R100 initialization should be used at the cost of increased yet still manageable computations.
- BCA and KMM are overall extremely fast and can handle datasets of size $n = 10^3$ and up without difficulty. KMM is slightly less accurate than BCA in terms of optimization performance (relative error between $10^{-3}$ and $10^{-4}$) and Rand index.
- 2X is computationally costly and fairly inaccurate when used on its own, that is, with an arbitrary starting point. It largely improves the accuracy of KMM solutions but not of BCA solutions. It is mostly beneficial in small to moderate dimension $n$.
- HUB and REC are not sufficiently accurate to be used on their own but they provide good starting points to more sophisticated matching methods. HUB uses data more extensively than REC and yields slightly better performance.
- For moderate to large $n$, EM shows poor performance in both computations (due to the evaluations of matrix permanents) and optimization. Its performance is satisfactory for $n \leq 50$, possibly because of the BCA initialization.
- ILP and KUR-BQP are only computationally feasible in very small samples ($n \leq 10$ or so). In this setup they virtually always find the global minimum of (1).
- KUR-ILP is relatively fast (it solves (1) for $n = 1000$ in 50 sec) but not highly accurate (relative error between 3% and 5%). LCVQ is both faster and far less accurate: it solves (1) for $n = 1000$ in 13 sec but has relative error in (0.2, 0.3) for all $n$.
- COP-KM and MPC-KM have very similar profiles in computation time and optimization accuracy. Their relative error goes from 0.2–0.3 for $n = 5$ to 0.05–0.06 for $n = 100$. They are not able to handle large datasets (at least not in their R implementation) as their computations stall for $n \geq 200$. CCLS only performs reasonably well for $n \leq 10$. Its Rand index and relative error deteriorate quickly as $n$ increases and its computations time out for $n \geq 500$.

4. **Application to fMRI Data**

In this section we harness the matching problem (1) and its proposed solutions to analyze resting-state functional magnetic resonance imaging (rs-fMRI) data, the goal being to explore the dynamic functional connectivity (DFC) of the brain. In short, functional connectivity (FC) relates to the integration of brain activity, that is, how distant brain regions coordinate their activity to function as a whole. The dynamic nature of FC, in particular its dependence on factors such as task-related activity, psychological state, and cognitive processes, is well established in neuroimaging research (e.g., Chang and Glover 2010; Handwerker et al. 2012; Hutchison et al. 2013).

The present analysis aims to extract measures of DFC from individual subject data and match these measures across subjects to uncover common patterns and salient features. The data under consideration are part of the ABIDE preprocessed data (Craadock et al. 2013), a large corpus of rs-fMRI measurements recorded from subjects diagnosed with autism spectrum disorder and from control subjects. These data and detailed descriptions are available at [preprocessed-connectomes-project.org/abide/](http://preprocessed-connectomes-project.org/abide/). We selected the following preprocessing options: Connectome Computation System (CCS) pipeline, spatial averaging over 116 regions of interest (ROI) defined by the AAL brain atlas, bandpass temporal filtering, no global signal regression. For simplicity, we only used data from control subjects and discarded data that did not pass all quality control tests. This resulted in $n = 308$ subjects with fMRI time series of average length about 200 scans (SD = 62).

**Subject-level analysis.** Vector autoregressive (VAR) models are widely used to assess FC in fMRI data (Valdés-Sosa et al. 2005; Friston, Moran, and Seth 2013; Ting et al. 2018). Here we represent the fMRI time series of a subject by a piecewise VAR model of order 1:

$$
y_t = A_t y_{t-1} + b_t + \varepsilon_t \quad (1 \leq t \leq T)
$$

where $y_t$ is an fMRI measurement vector of length 116, $A_t$ an unknown regression matrix encoding FC dynamics, $b_t$ an unknown baseline vector, and $\varepsilon_t$ a random noise vector with multivariate normal distribution $N(0, Q_t)$. The $A_t$ are assumed sparse, reflecting the fact that only a small number of ROIs at time $t - 1$ are predictive of ROI activity at time $t$. The model parameters $(A_t, b_t, Q_t)$ are assumed piecewise constant with few change points, indicating that FC states persist for some time (say, between 5 and 50 scans) before the brain switches to a different FC state.

For each subject, the task at hand is to simultaneously detect change points in (16) and estimate $(A_t, b_t)$ over the associated time segments. ($Q_t$ is of secondary importance here and can be ignored). The sparse group fused lasso (SGFL) approach of Degas (2021) is designed for this purpose. To simplify the task of determining a suitable range for the SGFL regularization parameters and calculating regularization paths, we employ the two-step procedure of this article. The first step detects change points via the group fused lasso (e.g., Bleakley and Vert 2011); the second step recovers sparse estimates of the $A_t$ separately on each segment via the standard lasso (Tibshirani 1996).

After fitting the regularization paths, a single lasso estimate $\hat{(A_t, b_t)}$ is selected for each segment by the Akaike Information Criterion. Among all generated model segmentations, we retain the one with the most segments satisfying the following criteria:
(i) \textit{length}: the segment must have at least five scans, (ii) \textit{goodness of fit}: the lasso fit must have a deviance ratio at least 0.3, and (iii) \textit{distinctness}: the parameter estimate $\hat{A}_k$ for the segment must have at least 10\% relative difference with estimates of other selected segments. To facilitate interpretation and remove noisy components, 10 segments at most are retained per subject.

\textbf{Group-level analysis.} Following the subject-level analysis, a set of change points and associated model parameter estimates is available for each subject, say $\{(\hat{A}_{ik}, \hat{b}_{ik}, \hat{T}_{ik}) : k \in [m_i]\}$ with $\hat{T}_{ik}$ the $k$th change point and $m_i$ the number of segments for the $i$th subject ($1 \leq i \leq n$). The regression matrices $\hat{A}_{ik}$ provide informative FC measures and could in principle be used for group-level comparisons. They are however, highly sparse and matching them using the squared Euclidean distance of problems (1)–(2) does not produce sensible results. We thus calculate the empirical correlation matrices on each segment $\{\hat{T}_{ik}, \ldots, \hat{T}_{i(k+1) - 1}\}$ and take them as inputs for the group analysis. After discarding correlation matrices based on short segments (10 scans or less) to increase estimation accuracy, we extract the lower halves of the remaining matrices and obtain a set $\{x_{ik} : 1 \leq i \leq 306, k \in [m_i]\}$ of 1801 correlation vectors of size $p = 116 \times 115/2 = 6670$. The number $m_i$ of vectors per subject varies in the range $[1, 10]$ with an average of 5.88 (SD = 1.77). The unbalanced matching problem (2) is then solved for $K \in \{10, 20, \ldots, 100\}$ using a generalized version of the BCA Algorithm 2. Based on the inspection of the cluster centers and cluster sizes, we retain the matching based on $K = 100$ clusters. With this choice, cluster sizes are in the range $[12, 28]$ (mean = 18.01, SD = 4.16). Smaller values of $K$, say $K \geq 50$, would be equally fine for data exploration.

Figure 3 displays the 100 resulting cluster centers, that is, the average correlation matrices of the clusters. For easier visualization and interpretation, the ROI-level correlations are aggregated into six well established \textit{resting state networks} (RSN): the attentional network (26 ROIs), auditory network (6 ROIs), default mode network (32 ROIs), sensorimotor network (12 ROIs), subcortical network (8 ROIs), and visual network (14 ROIs). A list of the ROI names and associated RSNs is given in the supplementary materials. Note that some ROIs do not belong to any known functional networks while others are recruited in two networks. The visual network and auditory network have strong intracorrelation (0.59 and 0.64 on average across cluster centers, respectively, not including TPOsup in the auditory network). The subcortical network and sensorimotor network show moderate internal correlation (0.51 on average each). The default mode and attentional networks comprise more ROIs and are usually less correlated (0.36 and 0.40 on average, respectively). The hippocampus (HIP), parahippocampal gyrus (PHG), and amygdala (AMYG) cluster together fairly strongly (average correlation 0.53). Applying community detection algorithms to each cluster center with the R package igraph, we noticed that ROIs from the visual network are virtually always in the same community; the same holds true for the subcortical network. The strongest correlations found between RSNs are the following: auditory–sensorimotor (0.38 on average across clusters) attentional–default mode (0.36), attentional–sensorimotor (0.36), and sensorimotor–visual (0.35).

Figure 4 shows interesting examples of average correlation matrices (cluster centers) at the ROI level. Cluster 5 shows strong positive correlation within the auditory, subcortical, and visual networks, and in the small groups (HIP, PHG, AMYG), with strong connections to the attentional network.
Figure 4. rs-fMRI data analysis. Examples of cluster centers (averages correlation matrices) derived from matching individual correlation matrices across subjects. Each displayed matrix corresponds to a cluster of 14–23 subjects.

(CRUS1, CRUS2), and (CB3–CB6, VERMIS1–VERMIS7). ROL has moderate to strong negative correlation with CRUS1, CRUS2 and regions from the subcortical network (dark blue stripe toward the top and left) and strong positive correlation with PoCG, SMG (sensorimotor) and HES, STG (auditory). The auditory and sensorimotor networks have moderate to strong
positive correlation. Cluster 14 shows clear blocking structure along the diagonal (correlation within RSN) as well as anticorrelation patterns between CAU, PUT, PAL, THA (subcortical) and ROL, PoCG (sensorimotor), PCL (sensorimotor); and between PCG (default mode) and PreCG (sensorimotor), ROL, PoCG (sensorimotor), PCL (sensorimotor). Community detection reveals three large and heterogeneous communities (sizes 43, 40, 36). Cluster 19 displays moderate to strong negative correlation (-0.55, -0.25) between IPL, SMG, ROL, CB10.R on the one hand and about 40 other ROIs on the other. The alternating clear and dark lines in cluster 27 reveal lateralized anticorrelation patterns between ROIs in the attentional network on the left side of the brain with most other ROIs in the brain. Cluster 42 shows two roughly uncorrelated blocks, a very large one with strong intracorrelation and a smaller one (CRUS, CB, VERMIS) with weaker intracorrelation. Cluster 88 displays a checked correlation structure with strong anticorrelation between (CRUS, CB, VERMIS) and the rest of the brain.

**Summary of the data analysis.** The data analysis has established that the matching approach (1)–(2) provides scientifically meaningful insights into DFC at the group level. By inspecting the cluster centers (average correlation matrices) produced by the matching process, one recovers large-scale patterns consistent with neuroscientific knowledge. For example, known resting state networks are clearly reflected in the blocking structure of the cluster centers (see Figure 4). But the cluster centers can also generate new insights and hypotheses. For example, the Heschl gyrus (HES) is not systematically included in the auditory network but, according to our analysis, it should. Similarly, the ROI TPOsup (temporal lobe: superior temporal gyrus), although it is near to or part of the auditory cortex, has shown only weak correlation with the other ROI of the auditory network, Superior temporal gyrus (STG). These elements may lead to a more nuanced understanding of the auditory network. Other remarkable findings include the strong anticorrelations found between the Rolandic operculum (ROL), the cerebellum (CER) and the vermis (VERMIS) on the one hand and (a large part of) the rest of the brain on the other. Importantly, by design, each of the clusters formed by the matching process highlights commonalities between subjects and not within subjects. This is in contrast with unconstrained clustering methods (e.g., $k$-means clustering) whose clusters may consist in (vectors from) a small number of or even a single subject in extreme cases.

5. Discussion

This research has investigated the task of matching feature vectors across many datasets in a one-to-one fashion. This task was formulated as a multidimensional assignment problem with decomposable costs (MDADC). To approximately solve this NP-hard combinatorial problem, we have developed algorithms based on $k$-means clustering, block coordinate ascent (BCA), and pairwise interchange heuristics, as well as several initialization methods. We have also taken a probabilistic approach to the matching problem, leading to a novel EM algorithm for Gaussian mixture models with permutation constraints. The main algorithms of the paper, $k$-means matching (KMM) and BCA, have iteration complexity $O(m^3n + m^2np)$ where $m$ is the number of data classes, $n$ the number of datasets, and $p$ the dimension of the feature vectors. They require few iterations to converge in practice and do not require precalculating or storing assignment costs. Thanks to these properties (in particular, the roughly linear computational complexity in $n$ and $p$), these algorithms are ideally suited for large-scale matching problems involving many datasets and possibly high-dimensional feature vectors.

While the article focuses on one-to-one feature matching with datasets of equal size, the proposed algorithms can be extended to match unbalanced datasets and/or to assign feature vectors to a variable number of clusters as in problem (2). This problem retains an essential characteristic of one-to-one matching, namely that each dataset contributes at most one vector per cluster. This guarantees that clusters capture common patterns across datasets and not within.

In the numerical study, the KMM and BCA algorithms show excellent optimization performance and computation time. In particular, they outperform all benchmark methods from the literature and can handle large collections of data. As suggested by theory, the BCA algorithm performs slightly better than KMM. The pairwise interchange heuristic can enhance KMM as a post-processing step (at a hefty computational price) but not BCA. The EM algorithm performs poorly throughout the study in large part because the allocation probabilities of data vectors to mixture components are almost invariably calculated as 0 or 1. This may in turn be caused by the relatively high dimension of the data ($p = 64$), the short tails of the normal distribution, and/or error in covariance estimation. Possible solutions could be to impose a diagonal structure on covariance estimates or to consider (mixtures of) distributions with heavier tails such as multivariate $t$-distributions. The slowness of the EM computations may be remedied by calculating a small fixed number of most likely allocations (Murty 1968) rather than computing them all via matrix permanents.

The analysis of the ABIDE fMRI data demonstrates that the proposed feature matching approach has strong potential for producing interpretable clusters of neuroimaging biomarkers at the group level. In contrast to unsupervised clustering, one-to-one feature matching is guaranteed to produce clusters that reflect variations between subjects and not within. While feature matching was employed in our analysis for data exploration, this technique could also be used in a more principled way as a preliminary step to disentangle association ambiguities between biomarkers and/or to stratify subjects into small, homogenous groups prior to a group-level analysis. Such matching-based approach could be for example compared to the consensus clustering strategy of Rasero et al. (2019).

Possible extensions and future work.

- **Weighted (squared) Euclidean distance.** The squared Euclidean distance in (1) can be generalized to a weighted squared Euclidean distance $\|x\|^2_w = x^tWx$ with $W \in \mathbb{R}^{p \times p}$ a positive semidefinite matrix. Decomposing $W$ as $L^tL$ (e.g., by Cholesky decomposition), it suffices to premultiply each matrix $X_i$ by $L$ to formulate an equivalent problem (1) using the unweighted (squared) Euclidean distance.

- **Metaheuristics.** It would be worthwhile investigating whether metaheuristics such as tabu search or iterated local search...
(e.g., Gendreau and Potvin 2019) can substantially improve the optimization performance of the BCA and KMM algorithms.

- **Dissimilarity functions.** Taking the squared Euclidean distance for the dissimilarity function $d$ in the general MDADC (3)–(4) enables fast and scalable algorithms with low memory footprint. However, in some applications, this choice of $d$ may not adequately capture differences between feature vectors and/or may not be robust to noise. If for example the cityblock distance $\| \cdot \|$ or Euclidean distance $\| \cdot \|_2$ were a better choice, a reasonable approach would be to relaxations of the MDADC (1) that yield tight lower bounds for worst case performances (e.g., Gutin, Goldengorin, and Leibniz formula for determinants, but without the permutation signature on $\mathcal{E}_k(\theta)$). Expand the conditioning on $I_{ikl} = 1$ to all other indicator variables $I_{ikl'}$ for $(k', l') \neq (k, l)$, and express this conditioning as $I_{l1σ(1)} = \cdots = I_{lσ(m)} = 1$ for some $σ \in \mathcal{S}_m$ such that $σ(k) = l$ (note that $(k' l') \notin \mathcal{E}_m$ is a random permutation matrix associated with a random permutation $σ$). We have also used the fact that $P_{\theta}(I_{l1σ(1)} = 1, \ldots, I_{lσ(m)} = 1) = 1/m!$ for all $σ \in \mathcal{S}_m$.

Formula (18) can be conveniently expressed with matrix permanents. The permanent of a square matrix $A = (a_{ij})$ of dimension $m \times m$ is defined as $\text{per}(A) = \sum_{i_1, \ldots, i_m} a_{i_1 j_1} \cdots a_{i_m j_m}$. Write $A_1 = (a_{i j}) = (\varphi(x_{ik}; \hat{\mu}_i, \hat{\Sigma}_l) \in \mathbb{R}^{m \times m}$ and $A_2 = (a_{k l'}) = (\varphi(x_{ik}; \hat{\mu}, \hat{\Sigma}_l) \in \mathbb{R}^{(m - 1) \times (m - 1)}$. By definition of the permanent and because only the permutations $σ$ such that $σ(k) = l$ are considered, the sum-product in (18) equals $a_{k l'}^2 \text{per}(A_1^{-1}(k l'))$. It also holds that $\sum_{k=1}^m E(I_{ikl} | X_1) = 1$. Summing (18) for $k = 1, \ldots, m$ and exploiting the two previous results, we obtain that $c_i / m! = \sum_{k=1}^m a_{ikl} \text{per}(A_1^{-1}(k l)) = 1$. Finally, noting that $\text{per}(A_1) = \sum_{k=1}^m a_{ikl} \text{per}(A_1^{-1}(k l))$ (basic properties of permanents), it follows that $c_i / (m! - 1) = 1 / \text{per}(A_1)$. Therefore

$$P_{\theta}(I_{ikl} = 1 | X_i) = a_{ikl} \text{per}(A_1^{-1}(k l)) / \text{per}(A_1).$$

The permanent of a matrix has a very similar expression to the Leibniz formula for determinants, but without the permutation signatures $\pm 1$. It is however, far more expensive to compute: efficient implementations have complexity $O(2^m m^2)$ (Rysier 1963) or $O(2^m m)$ (Nijenhuis and Wilf 1978). Stochastic approximation methods running in polynomial time (e.g., Jerrum, Sinclair, and Vigoda 2004; Kuck et al. 2019) and variational bounds (see Uhlmann 2004, and the references therein) are also available. Given that (18) must be evaluated for $nm^2$ values of $(i, k, l)$, and accounting for the computation of the matrices $A_1(i \in [n])$ (e.g., Press et al. 2007, Chap. 16.1), the E step has overall complexity at least $O(2^m m^2 + m^{3/2} \cdot m^2)$.

The evaluation of permanents requires precautions to avoid numerical underflow. Indeed, the density values $\varphi(x_{ik}; \hat{\mu}_i, \hat{\Sigma}_l)$ are often very small and multiplying them in (18) may quickly lead to numerical zeros. Preconditioning greatly helps in this regard: by the properties of the permanent, multiplying the rows and columns of $A_1$ by nonzero numbers has no effect on (18) as these multiples cancel out between the numerator $a_{ikl} \text{per}(A_1^{-1}(k l))$ and denominator $\text{per}(A_1)$. One can exploit this by alternatively rescaling the rows and columns of $A_1$ by their sums. Provided that $A_1$ is a positive matrix, this scheme converges to a doubly stochastic matrix (Sinkhorn 1964) that in practice often has at least one “non-small” entry in each row and each column.
M step. By standard least square calculations, the updated estimate $	heta^+ = [(\mu_i^+, \Sigma_i^+): 1 \leq l \leq m]$ is
\[
\begin{align*}
\mu_i^+ &= \frac{1}{n} \sum_{k=1}^{n} \sum_{m=1}^{m} p_{\theta}(i|X_k) x_{ik} \\
\Sigma_i^+ &= \frac{1}{n} \sum_{k=1}^{n} \sum_{m=1}^{m} p_{\theta}(i|X_k) (x_{ik} - \mu_i^+) (x_{ik} - \mu_i^+)' \\
\end{align*}
\]
with $p_{\theta}(i|X_k) = 1(X_i)$ given by (19). The fact that $\sum_{k=1}^{n} p_{\theta}(i|X_k) = 1(X_i)$ for all $(i, l)$ was used to simplify (20). If the variances $\Sigma_1, \ldots, \Sigma_m$ are assumed equal, common standard values should be $(1/m) \sum_{i=1}^{n} \Sigma_i^+$.

Log-likelihood. The log-likelihood of the observed data is given by
\[
\log L(\hat{\theta}) = \sum_{i=1}^{n} \log \left( \frac{1}{m!} \sum_{\sigma \in S_m} \prod_{i=1}^{m} \psi(x_{ik}; \hat{\mu}(k), \hat{\Sigma}(k)) \right). 
\]
It is simply the sum of the logarithms of the permanents of the matrices $A_i = (\psi(x_{ik}; \hat{\mu}_i, \hat{\Sigma}_i))$ defined earlier. Since these permanents are calculated in the E step, there is essentially no additional cost to computing the log-likelihood.

The EM algorithm for model (13) is sketched in Algorithm 4.

### Algorithm 4 EM for Constrained Gaussian Mixture

**Require:** $X_1, \ldots, X_n \in \mathbb{R}^{p \times m}, \mu_1, \ldots, \mu_m \in \mathbb{R}^p, \Sigma_1, \ldots, \Sigma_m \in \mathbb{R}^{p \times p}$

1: $\theta^0 \leftarrow \{ (\mu_i, \Sigma_i): 1 \leq l \leq m \}$
2: for $t = 0, 1, \ldots$
3: \hspace{1em} Perform Choleski decomposition $\Sigma_i = L_i^T L_i$ with $L_i$ lower triangular ($1 \leq l \leq m$)
4: \hspace{1em} for $i = 1, \ldots, n$
5: \hspace{2em} $a_{ik} \leftarrow \frac{1}{m} \sum_{j=1}^{m} (x_{ij} - \mu_j) (x_{ij} - \mu_j)'$
6: \hspace{1em} for $k = 1, \ldots, m$
7: \hspace{2em} for $l = 1, \ldots, m$
8: \hspace{3em} Alternatively rescale rows and columns of $A_i^{-1}(k,l)$ to sum to 1
9: \hspace{3em} Calculate per($A_i^{-1}(k,l)$) with Ryser’s inclusion-exclusion formula
10: \hspace{2em} $p_{ik} \leftarrow a_{ik} \text{per}(A_i^{-1}(k,l))$
11: end for
12: end for
13: $c_i \leftarrow 1 \sum_{k=1}^{n} \sum_{l=1}^{m} \frac{p_{ik}}{c_l}$ (1 \leq k, l \leq m) [class membership probability]
14: $\ell^t \leftarrow \sum_{i=1}^{n} \log c_i$ [log-likelihood]
15: for $i = 1, \ldots, m$
16: \hspace{1em} $\mu_i \leftarrow \frac{1}{n} \sum_{k=1}^{n} \sum_{l=1}^{m} w_{ik} x_{ik}$
17: \hspace{1em} $\Sigma_i \leftarrow \frac{1}{n} \sum_{k=1}^{n} \sum_{l=1}^{m} \sum_{m=1}^{m} w_{ik} x_{ik} (x_{ik} - \mu_i) (x_{ik} - \mu_i)'$
18: end for
19: $\theta^{t+1} \leftarrow \{ (\mu_i, \Sigma_i): 1 \leq i \leq m \}$
20: end for

**Supplementary Materials**

**Details on data analysis and theoretical aspects:** List of regions of interest and of resting state networks considered in the fMRI data analysis (Section 4). Proof of global optimality result in the convex relaxation of the matching problem (Appendix B). File: supplement.pdf.

**Acknowledgments**

The author thanks the Editor and two anonymous reviewers for their helpful comments. He also thanks Vince Lyzinski for his valuable insights on convex relaxation.
McLachlan, G., and Peel, D. (2000), *Finite Mixture Models*, Wiley Series in Probability and Statistics: Applied Probability and Statistics, New York: Wiley-Interscience. [201]

Munkres, J. (1957), “Algorithms for the Assignment and Transportation Problems,” *Journal of the Society for Industrial and Applied Mathematics*, 5, 32–38. [199]

Murty, K. G. (1968), “An Algorithm for Ranking all the Assignments in Order of Increasing Cost,” *Operational Research*, 16, 682–687. [208]

Natu, S., Date, K., and Nagi, R. (2020), “GPU-Accelerated Lagrangian Heuristic for Multidimensional Assignment Problems with Decomposable Costs,” *Parallel Computing*, 97, 102666. [197,202]

Nijenhuis, A., and Wilf, H. S. (1978), *Combinatorial Mathematics*, New York-London: Academic Press, Inc. [209]

Oliveira, C. A. S., and Pardalos, P. M. (2004), “Randomized Parallel Algorithms for the Multidimensional Assignment Problem,” *Applied Numerical Mathematics*, 49, 117–133. [200]

Pananjady, A., Wainwright, M. J., and Courtade, T. A. (2018), “Linear Regression with Shuffled Data: Statistical and Computational Limits of Permutation Recovery,” *IEEE Transactions on Information Theory*, 64, 3286–3300. [198]

Pardalos, P. M., and Pitsoulis, L. S., eds. (2000), *Nonlinear Assignment Problems*, vol. 7 of Combinatorial Optimization, Dordrecht: Kluwer Academic Publishers. [197]

Pelleg, D., and Baras, D. (2007), “K-means with Large and Noisy Constraint Sets,” in *Proceedings of the 18th European Conference on Machine Learning (ECML)*, Springer, pp. 674–682. [197,202]

Pierskalla, W. P. (1968), “The Multidimensional Assignment Problem,” *Operational Research*, 16, 422–431. [196,197]

Poore, A. P., and Rijavec, N. (1993), “A Lagrangian Relaxation Algorithm for Multidimensional Assignment Problems Arising from Multitarget Tracking,” *SIAM Journal on Optimization*, 3, 544–563. [197]

Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P. (2007), *Numerical Recipes: The Art of Scientific Computing* (3rd ed.), Cambridge: Cambridge University Press. [209]

Rand, W. M. (1971), “Objective Criteria for the Evaluation of Clustering Methods,” *Journal of the American Statistical Association*, 66, 846–850. [201]

Rasero, J., Diez, I., Cortes, J. M., Marinazzo, D., and Stramaglia, S. (2019), “Connectome Sorting by Consensus Clustering Increases Separability in Group Neuroimaging Studies,” *Nature Neuroscience*, 3, 325–343. [208]

Rezaatofghi, S. H., Milan, A., Zhang, Z., Shi, Q., Dick, A., and Reid, I. (2015), “Joint Probabilistic Data Association Revisited,” in *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 3047–3055. [196]

Robertson, A. J. (2001), “A Set of Greedy Randomized Adaptive Local Search Procedure (GRASP) Implementations for the Multidimensional Assignment Problem,” *Computational Optimization and Applications*, 19, 145–164. [197,200]

Ryser, H. J. (1963), *Combinatorial Mathematics*, vol. 14 of The Carus Mathematical Monographs, New York: Wiley. [209]

Shalom, M., Wong, P. W. H., and Zaks, S. (2010), “On-line Maximum Matching in Complete Multipartite Graphs with Implications to the Minimum ADM Problem on a Star Topology,” in *Structural Information and Communication Complexity*, vol. 5869 of Lecture Notes in Computer Science, pp. 281–294, Berlin: Springer. [196]

Shental, N., Bar-hillel, A., Hertz, T., and Weinshall, D. (2004), “Computing Gaussian Mixture Models with EM Using Equivalence Constraints,” in *Advances in Neural Information Processing Systems*, vol. 16, MIT Press, pp. 465–472. [197,201]

Sinkhorn, R. (1964), “A Relationship Between Arbitrary Positive Mmatrices and Doubly Stochastic Matrices,” *Annals of Mathematical Statistics*, 35, 876–879. [209]

Spieksma, F., and Woeginger, G. (1996), “Geometric Three-Dimensional Assignment Problems,” *European Journal of Operational Research*, 91, 611–618. [197]

Tauer, G., and Nagi, R. (2013), “A Map-reduce Lagrangian Heuristic for Multidimensional Assignment Problems with Decomposable Costs,” *Parallel Computing*, 39, 653–668. [197,202]

Tibshirani, R. (1996), “Regression Shrinkage and Selection via the Lasso,” *Journal of the Royal Statistical Society*, Series B, 58, 267–288. [205]

Ting, C. M., Ombao, H., Samdin, S. B., and Salleh, S. H. (2018), “Estimating Dynamic Connectivity States in fMRI Using Regime-Switching Factor Mmodels,” *IEEE Transactions on Medical Imaging*, 37, 1011–1023. [205]

Uhlmann, J. K. (2004), “Matrix Permanent Inequalities for Approximating Joint Assignment Matrices in Tracking Systems,” *Journal of the Franklin Institute*, 341, 569–593. [209]

Valdés-Sosa, P. A., Sánchez-Bornot, J. M., Lage-Castellanos, A., Vega-Hernández, M., Bosch-Bayard, J., Melie-García, L., and Canales-Rodríguez, E. (2005), “Estimating Brain Functional Connectivity with Sparse Multivariate Autoregression,” *Philosophical Transactions of the Royal Society B*, 360, 969–981. [205]

Vogelstein, J. T., Conroy, J. M., Lyzinski, V., Podrazik, L. J., Kratzer, S. G., Harley, E. T., Fishkind, D. E., Vogelstein, R. J., and Priebe, C. E. (2015), “Fast Approximate Quadratic Programming for Graph Matching,” *PloS One*, 10, 1–17. [196]

von Neumann, J. (1953), “A Certain Zero-Sum Two-Person Game Equivalent to the Optimal Assignment Problem,” in *Contributions to the Theory of Games*, vol. 2, Annals of Mathematics Studies, no. 28, pp. 5–12, Princeton, NJ: Princeton University Press. [210]

Wagstaff, K., Cardie, C., Rogers, S., and Schrödl, S. (2001), “Constrained k-means Clustering with Background Knowledge,” in *Proceedings of the 18th International Conference on Machine Learning (ICML)*, pp. 577–584. [197,201,202]

Wang, L., Liu, T., Wang, G., Chan, K. L., and Yang, Q. (2015), “Video Tracking Using Learned Hierarchical Features,” *IEEE Transactions on Image Processing*, 24, 1424–1435. [196]

Wright, S. J. (2015), “Coordinate Descent Algorithms,” *Mathematical Programming*, 151, 3–34. [199]