Block rearranging elements within matrix columns to minimize the variability of the row sums

Boudt, K.M.R.; Jakobsons, Edgars; Vanduffel, Steven

published in
4OR
2018

DOI (link to publisher)
10.1007/s10288-017-0344-4

document version
Publisher's PDF, also known as Version of record

4OR-Q J Oper Res (2018) 16:31–50
https://doi.org/10.1007/s10288-017-0344-4

RESEARCH PAPER

Block rearranging elements within matrix columns to minimize the variability of the row sums

Kris Boudt[1,2] · **Edgars Jakobsons**[3] · **Steven Vanduffel**[1]

Received: 13 November 2016 / Revised: 31 January 2017 / Published online: 2 March 2017

Abstract Several problems in operations research, such as the assembly line crew scheduling problem and the k-partitioning problem can be cast as the problem of finding the intra-column rearrangement (permutation) of a matrix such that the row sums show minimum variability. A necessary condition for optimality of the rearranged matrix is that for every block containing one or more columns it must hold that its row sums are oppositely ordered to the row sums of the remaining columns. We propose the block rearrangement algorithm with variance equalization (BRAVE) as a suitable method to achieve this situation. It uses a carefully motivated heuristic—based on an idea of variance equalization—to find optimal blocks of columns and rearranges them. When applied to the number partitioning problem, we show that BRAVE outperforms the well-known greedy algorithm and the Karmarkar–Karp differencing algorithm.

Keywords Assembly line crew scheduling · Greedy algorithm · Rearrangements · k-Partitioning · Karmarkar–Karp differencing algorithm

Mathematics Subject Classification 90B35 · 90B90 · 90C27 · 97M40 · 90C59

✉ Steven Vanduffel
steven.vanduffel@vub.ac.be

Kris Boudt
kris.boudt@vub.ac.be

Edgars Jakobsons
edgars.jakobsons@math.ethz.ch

1 Vrije Universiteit Brussel (VUB), Brussels, Belgium

2 Vrije Universiteit Amsterdam, Amsterdam, The Netherlands

3 ETH Zürich, Zürich, Switzerland

1 Introduction

In this paper, we study the **rearrangement problem** of finding an intra-column rearrangement of an $n \times d$ matrix, such that the row sums show minimum variability. This problem is observationally equivalent to the problem of optimal assembly line crew scheduling (ALCS); see Hsu (1984). The most obvious way of solving the problem would be by brute force enumeration. However, as there are $(n!)^{d-1}$ rearranged matrices to consider, doing so is not feasible unless n and d are small. In fact, whenever $d > 2$, the problem is NP-hard (Coffman and Yannakakis 1984), explaining why it is typically dealt with using heuristic approaches. In this regard, a recent algorithm applicable to this purpose is the so-called rearrangement algorithm (RA) of Puccetti and Rüschendorf (2012). The RA aims at solving the rearrangement problem by rearranging[1] the elements within the columns of the matrix so that each column becomes oppositely ordered to the sum of *all other* columns. This heuristic has a strong theoretical support by the fact that a necessary condition for row sums to have minimum variance is that the elements in each column are oppositely ordered to the row sums of the other columns. A related, but less performant algorithm can be found in Coffman and Yannakakis (1984). These authors order the elements of the columns in the opposite order to the row sums of *all preceding* columns, an approach[2] that is also inherent in Graham's seminal list scheduling heuristic;[3] see Graham (1966).

The quality of the approximation obtained by the RA can be further improved by rearranging blocks of columns. Indeed, if any block of two or more columns in the output matrix of the RA can be further rearranged so that its row sums become oppositely ordered to the row sums of the remaining columns, then doing so improves the solution. This basic insight reveals that the RA can be improved by considering block rearrangements. The idea of using block rearrangements is not entirely new. It was first mentioned in Bernard et al. (2015) (see their Remark 4.1) and further used[4] in Bernard et al. (2017) under the name block RA (Remark 3.3). However, a precise description of the approach as well as a study of its properties and its potential use in solving problems in operations research is missing in the literature. Our main contributions can be summarized as follows:

First, we propose a carefully motivated heuristic to find an approximately optimal block in each iteration and label this approach as the block rearrangement algorithm with variance equalization (BRAVE). We show that BRAVE can be seen as an extension to higher dimensions of the seminal Karmarkar–Karp (KK) differencing algorithm that was introduced to deal with the so-called number partitioning problem (Karmarkar

[1] Rearranging refers to the act of permuting (swapping) elements.

[2] In Bernard et al. (2017) it is shown that for some particular type of matrices this approach makes it possible to obtain after d − 1 steps (rearrangements) the situation in which all columns are oppositely ordered to the sum of all others. This development allows to obtain approximations for the minimum variance of a sum of (scaled) Bernoulli distributions in a fast way.

[3] In Graham (1966), the so-called parallel machine scheduling problem is solved by iteratively assigning subsequent jobs to the machine whose current completion time is minimum.

[4] The use of block rearrangements is consistent with the notion of Σ-countermonotonic matrices, as developed in Section 3.4 of Puccetti and Wang (2015); see in particular Definition 3.8 and Theorem 3.8 herein.

and Karp 1982). When applied to number partitioning, we show that BRAVE outperforms the KK differencing algorithm.

Second, we compare the performance of the different rearrangement algorithms for various set-ups of matrices, i.e., for various levels of dimensionality of the rows and columns as well as for different levels of heterogeneity among the matrix elements. We show that BRAVE greatly reduces the variability during the first series of iterations. Specifically, using BRAVE as a presolver (i.e., using it for the first series of iterations) and then proceeding further with classical RA yields overall good results.

The rest of this paper is organized as follows. In the remainder of this section, we describe the rearrangement problem and discuss variability measures (objective functions) that are compatible with the idea of block rearrangements. In Sect. 2, we provide a necessary condition for optimal solutions of the rearrangement problem. In Sect. 3, we recall the existing rearrangement algorithms and propose BRAVE as a suitable algorithm to deal with the rearrangement problem. In Sect. 4, we describe its connection with the KK differencing algorithm and show that BRAVE outperforms KK for the number partitioning problem. The performance of the rearrangement algorithms is assessed in Sect. 5. Concluding remarks are given in Sect. 6.

1.1 The problem setup

The input for the rearrangement problem consists of d n-dimensional column vectors $X_1, X_2, \ldots, X_d \in \mathbb{R}^{n\times 1}$ that are combined into a matrix $X := (X_1, X_2, \ldots, X_d) \in \mathbb{R}^{n\times d}$. Denote by $X^\pi := (X_1^\pi, X_2^\pi, \ldots, X_d^\pi) \in \mathbb{R}^{n\times d}$ the resulting matrix after rearranging the n elements of the jth column ($j = 1, 2, \ldots, d$) by a permutation $\pi_j : \{1, 2, \ldots, n\} \to \{1, 2, \ldots, n\}$, i.e., $(X^\pi)_{ij} = (X)_{\pi_j(i)j}$, where $\pi := (\pi_1, \pi_2, \ldots, \pi_d)$ denotes the vector of the d permutations. Throughout, we will call the vector of permutations π a *rearrangement* and X^π a *rearranged matrix*. Denote the vector of the corresponding row sums by $S^\pi := \sum_{j=1}^d X_j^\pi$. This paper makes use of a particular kind of rearrangements, called *block rearrangements*. A block rearrangement consists in swapping entire rows within a block of columns (and not just elements within a given column). Specifically, let $\delta \in \{0, 1\}^d$ indicate by $\delta_j = 1$ the columns j to be rearranged, $j = 1, \ldots, d$. A rearrangement π' of a matrix X^π into $X^{\pi'}$ is called a block rearrangement if there exists a $\delta \in \{0, 1\}^d$ and a permutation $\pi^\delta : \{1, 2, \ldots, n\} \to \{1, 2, \ldots, n\}$ such that $\pi'_j(\cdot) = \pi_j(\pi^\delta(\cdot))$ for $\delta_j = 1$ and $\pi'_j = \pi_j$ for $\delta_j = 0$.

We formally state the **rearrangement problem** as

$$\min_\pi V\left(S^\pi\right), \tag{1}$$

where $V(\cdot)$ is a scalar-valued function that measures the *variability* of the vector S^π. In what follows we will sometimes call $V(\cdot)$ a variability measure.

The algorithms that we propose to deal with the rearrangement problem (1) are well motivated under a certain assumption on the variability measure $V(\cdot)$. To this end, we first recall the notion of majorization order. Hence, let $W, Z \in \mathbb{R}^n$, then we say that W is smaller than Z in the sense of the majorization order, denoted as $W \preccurlyeq_{\mathrm{m}} Z$, if

$$\sum_{i=1}^{n} w_{(i)} = \sum_{i=1}^{n} z_{(i)}$$

and for $k = 1, 2, \ldots, n-1$,

$$\sum_{i=1}^{k} w_{(i)} \leq \sum_{i=1}^{k} z_{(i)}$$

where the subscript (i) is used to denote the ith largest element of a vector (decreasing order statistics). In words, W is majorized by Z if its components are more evenly spread out than the components of Z. Our basic assumption is that V respects majorization order; any such V is called *Schur-convex*.

Assumption 1 The function $V(\cdot)$ is *Schur-convex*, i.e., it holds that for any W, $Z \in \mathbb{R}^n$,

$$W \preccurlyeq_{\mathrm{m}} Z \quad \text{implies} \quad V(W) \leq V(Z).$$

Important examples of Schur-convex functions arise when V is of the form $V(S^\pi) = \sum_{i=1}^{n} f(S_i^\pi)$, where $f(\cdot)$ is a convex function; in particular, the variance is Schur-convex (rearrangements do not change the mean value of the row sums). Other examples are given by L-statistics (linear combinations of order statistics) with decreasing coefficients, thus also the maximum and (minus) the minimum are Schur-convex.[5] We refer to Marshall et al. (2011), Rüschendorf (2013), Boland and Proschan (1988) and Jakobsons and Wang (2016) for further details on the compatibility of rearrangement methods with Schur-convex objective functions.

Remark 1 It is possible to study rearrangement problems in a probabilistic setting. In this case, one interprets the matrix $X = (X_1, X_2, \ldots, X_d) \in \mathbb{R}^{n \times d}$ as a random vector; each column corresponds to a random variable and each row is a possible joint outcome. Furthermore, all outcomes occur with probability $1/n$. A rearrangement π thus yields a rearranged matrix X^π in which the variables X_j^π have the same (marginal) distribution function as the X_j $(j = 1, 2, \ldots, d)$, but their interdependence has changed, i.e., the joint distributions of $(X_1, X_2, \ldots, X_d)$ and $(X_1^\pi, X_2^\pi, \ldots, X_d^\pi)$ are different. In this paper, however, we stick to the matrix formulation. Nevertheless, we sometimes use terminology (such as the mean, variance and distribution) that strictly speaking is better adapted to the probabilistic formulation.

2 Characterization of optimal rearrangements

Let $X := (X_1, X_2, \ldots, X_d) \in \mathbb{R}^{n \times d}$ be a given matrix for which the vector S of the n row sums has variability $V(S)$. Under Assumption 1, it holds that, if there exists a rearrangement π of the X_j such that S^π becomes constant and thus equal to the mean, then it is the solution to the rearrangement problem (1).[6] In general, one cannot

[5] See also Section 2.3 in Bernard et al. (2015).

[6] In probabilistic terms, this specific situation corresponds to mixability of random variables, a concept that has recently been studied in Wang and Wang (2011, 2016).

expect to obtain a constant S^π and finding the optimal rearrangement π remains a challenging task.

However, it is clear that a candidate solution (i.e., a rearrangement π) can be improved, if one can find another rearrangement π' that yields a sum $S^{\pi'}$ that is smaller in majorization order than the sum S^π. At this point, we recall the well-known fact that for $W \in \mathbb{R}^n$ and $Z \in \mathbb{R}^n$ the minimizing rearrangement for their sum $W + Z$ with respect to the majorization order $\preccurlyeq_{\mathrm{m}}$ occurs when W and Z are *oppositely ordered*[7] (Day 1972). Hence, if the row sums of a block of columns are not oppositely ordered with the row sums of the remaining columns (complementary block), rearranging that block so that its row sums become oppositely ordered to those of the complementary block, will reduce the variability of the vector that contains the row sums of the matrix. The following proposition is thus proven.

Proposition 1 (Improvement of a solution by block rearrangement) *Let V be a variability measure satisfying* Assumption 1. *Suppose that for $X^\pi \in \mathbb{R}^{n\times d}$, there exists $\delta \in \{0,1\}^d$ such that $\sum_{j=1}^d \delta_j X_j^\pi$ and $\sum_{j=1}^d (1-\delta_j) X_j^\pi$ are not oppositely ordered. Let $\pi' = (\pi'_1, \ldots, \pi'_d)$ be a block rearrangement of π such that $\sum_{j=1}^d \delta_j X_j^{\pi'}$ and $\sum_{j=1}^d (1-\delta_j) X_j^{\pi'}$ are oppositely ordered. Then,*

$$V\left(\sum_{j=1}^d \delta_j X_j^{\pi'} + \sum_{j=1}^d (1-\delta_j) X_j^{\pi'}\right) \leq V\left(\sum_{j=1}^d \delta_j X_j^{\pi} + \sum_{j=1}^d (1-\delta_j) X_j^{\pi}\right).$$

Proposition 1 thus means that, each time a block is rearranged so that its row sums become oppositely ordered with the row sums of the complementary block, the variability of the vector containing all row sums decreases. It further follows that, if the objective V is of the form $V(S) = \sum_{i=1}^n f(s_i)$, where f is a strictly convex function, then the "$\leq$" in Proposition 1 can be replaced by a "$<$".[8] A minimum can then only be attained when all partitions into two blocks have the property that the partial rows sums corresponding to these blocks are oppositely ordered. Such a matrix is called **Σ-countermonotonic**; see also Section 3.4 in Puccetti and Wang (2015). The block rearrangement algorithms that we propose as heuristics for the rearrangement problem are in fact designed to find Σ-countermonotonic matrices. These algorithms are discussed further in the next section.

3 Rearrangement algorithms

Let X be a given matrix with a corresponding row-sum vector S. In order to find the optimal rearranged matrix X^π, we first consider the RA of Puccetti and Rüschendorf (2012), which rearranges single columns one-by-one, and then we discuss block rear-

[7] Vectors $W, Z \in \mathbb{R}^n$ are said to be oppositely ordered if $(w_j - w_i)(z_j - z_i) \leq 0$ for all $1 \leq i < j \leq n$.

[8] It is a direct consequence of the fact that unless W and Z have the same distribution, $W \preccurlyeq_{\mathrm{cx}} Z$ implies $E[f(W)] < E[f(Z)]$ for any strictly convex function f; See Theorem 3.A43 of Shaked and Shanthikumar (2007).

rangements. Similarly to Embrechts et al. (2013), all algorithms will start by applying a random permutation of the elements within each column.[9] Throughout the paper, we assume the convention that the columns of X are sorted in terms of decreasing variability, i.e., $V(X_i) \geq V(X_j)$, for all column numbers $1 \leq i \leq j \leq d$.[10]

3.1 Existing algorithms and some extensions

3.1.1 The rearrangement algorithm (RA)

The RA is a heuristic that aims at solving the rearrangement problem by iteratively rearranging the columns of the matrix one-by-one, so that each column becomes oppositely ordered to the sum of the other columns. In other words, the RA is not designed to achieve the necessary condition of optimality, i.e., a Σ-countermonotonic matrix (see also Proposition 1 and its discussion), but only a weaker condition that is obtained by restricting to vectors $\delta \in \{0, 1\}^d$ for which $\sum_{j=1}^{d} \delta_j = 1$.

Hence, the solution provided by the RA can be readily improved whenever there is a partition into two blocks with the property that the row sums of the first block of columns is not oppositely ordered to the row sums of the remaining columns (see Proposition 1). In order to address this concern, we consider algorithms that rearrange blocks of columns. In this regard, we use a d-dimensional binary vector $\delta \in \{0, 1\}^d$ to indicate which columns belong to the first block ($\delta_j = 1$), and which ones to the complementary block ($\delta_j = 0$). Each δ thus corresponds to a partition of the matrix X^π into two submatrices (blocks), denoted X^π_δ and $X^\pi_{1-\delta}$. Their corresponding row sums are denoted $S^\pi_\delta := \sum_{j=1}^{d} \delta_j X^\pi_j$ and $S^\pi_{1-\delta} := \sum_{j=1}^{d} (1 - \delta_j) X^\pi_j$.

3.1.2 Brute force Block RA

In the so-called brute force Block RA, we require that for all possible partitions into two blocks, the corresponding row sums of the two blocks are oppositely ordered (i.e., X^π is Σ-countermonotonic).[11] This implementation ensures that a further improvement of the objective using block rearrangements is not possible (see Proposition 1).

[9] Note that this *shuffling* step already leads to a substantial reduction in the variability among the row sums, compared to a sorted (co-monotonic) arrangement.

[10] Sorting of the columns is done to render our paper more transparent with respect to reproducibility of the numerical results and also to enable a more fair comparison between block rearrangement algorithms and the RA of Puccetti and Rüschendorf (2012) in the case of heterogeneous columns. In fact, the RA goes through the columns sequentially (starting with the first column and so on). If the first few columns show little variability, rearranging them would not have much effect on the variability of row sums. For the block rearrangement algorithms, this issue does not exist, as they do not apply rearrangements sequentially starting with the first column.

[11] To verify this condition, one needs to consider $(2^d - 2)/2 = 2^{d-1} - 1$ different partitions of the matrix into two blocks. The division by two is because for each partition δ, the corresponding $1 - \delta$ does not need to be considered. Furthermore, the cases in which all elements of δ are zero (resp. one), do not require consideration.

3.1.3 Random block RA (BRA)

In the brute force Block RA above, evaluating all possible vectors δ to verify the condition of Σ-countermonotonicity corresponds to considering $2^{d-1}-1$ partitions of the matrix (see Footnote 11). Doing so becomes infeasible for even moderate values of d (i.e. for $d > 15$). One solution, proposed in Bernard and McLeish (2016), is to consider a smaller (randomly generated) subset K of the set of all partitions. In Algorithm 1, we implement a related idea, in each iteration considering one random partition obtained by Bernoulli sampling, in which the probability for a matrix column to belong to either of the blocks is always 50% (uniform sampling over all partitions). We call this algorithm the (random) Block RA (BRA) in the following.

Algorithm 1: Pseudo-code of the procedure **random BRA** to minimize the variability V of the vector containing all row sums of a matrix X. Blocks are chosen randomly

1. Set target value a^*, and maximum number of iterations M;
2. Set π to a randomly chosen intra-column arrangement of the elements;
3. Set $a := V(S^\pi)$ with $S^\pi := \sum_{j_i}^{d} X_j^\pi$ and let $m = 0$;
4. **while** $a > a^*$ *and* $m < M$ **do**
   5. Generate a random vector $\delta \in \{0,1\}^d$ by Bernoulli sampling with equal probability for each column to be assigned to either block;
   6. Rearrange the rows of the submatrix X_δ^π so that the row sums S_δ^π become oppositely ordered to $S_{1-\delta}^\pi$, and compute $a := V(S_\delta^\pi + S_{1-\delta}^\pi)$;
   7. Set $m = m + 1$.

end

Note that the choice of the target value a^* in Algorithm 1 is somewhat arbitrary. Nevertheless, once a candidate solution to the rearrangement problem is obtained, one can always lower the target value based on the value a of the output matrix, and then use this matrix (and its corresponding rearrangement) further. A variation of this algorithm would correspond to a non-uniform sampling: using different probabilities for each column to be included in the first block, $\delta_j \sim \text{Bernoulli}(p_j)$, $j = 1, \ldots, d$. Another variation would be sampling the size N of the first block $P(N = i) = q_i, i = 1, \ldots, d-1$, and then selecting the first block uniformly over all blocks of size N (for example, $q_1 = 1$ and $q_i = 0, i \geq 2$, would lead to RA with a random column selection instead of sequential). However, these alternatives pose the practical problem of having to calibrate the block-selection probabilities *ex ante*; we do not investigate them further.

3.2 Block RA with variance equalization (BRAVE)

In the previous section, we pointed out that the number of possible partitions of the matrix columns is prohibitively large even for moderate values of d. Therefore, a method for choosing those partitions which yield the greatest improvement is desirable. Bernard and McLeish (2016) (see their block RA1) propose selecting the partition $\tilde{\delta}$ by

$$\tilde{\delta} = \underset{\delta \in \mathcal{P}}{\operatorname{argmax}}\, \phi\left(S_{\delta}^{\pi}, S_{1-\delta}^{\pi}\right), \tag{2}$$

where ϕ is Spearman's rank correlation and $\mathcal{P} \subset \{0,1\}^d$ is a randomly selected subset of all partitions, consisting of $|\mathcal{P}| = \min(2^{d-1} - 1, 512)$ partitions to keep the above problem tractable. This approach draws it inspiration from the fact that Spearman's correlation is equal to -1 for oppositely ordered vectors, so the rearranged matrix is Σ-countermonotonic if and only if Spearman's correlation is -1 for all partitions.

The above approach has two drawbacks. First, since the method needs to compute the rank correlation (requires sorting) for each considered partition, in practice, it cannot consider all possible partitions; and finding $\tilde{\delta}$ is rather time-consuming even with $|\mathcal{P}| = 512$. Second, using correlations ignores the scales of the partial row sums, so this method may choose a partition where the two partial row sums are of very different scales, resulting in little improvement in $V(S^{\pi})$.

We propose a new block selection rule, which is based on *variance equalization*. In particular, consider an initial matrix X^{π}. An optimal choice for the two blocks X_{δ}^{π} and $X_{1-\delta}^{\pi}$ would clearly occur when, *after* rearranging S_{δ}^{π} in an opposite order to $S_{1-\delta}^{\pi}$, it holds that $S_{\delta}^{\pi'} + S_{1-\delta}^{\pi'} = E[S]$, where π' denotes the new rearrangement.[12] A necessary condition to fulfill this condition is that, *before* rearranging, $S_{\delta}^{\pi} - E[S_{\delta}^{\pi}]$ is a rearrangement of $E[S_{1-\delta}^{\pi}] - S_{1-\delta}^{\pi}$. In particular, they need to have the same mean and variance. Since by construction their mean values are already matched, this motivates selecting an optimized partitioning vector δ^* so that the *second* moments of the distributions are matched as closely as possible. Under this approach of selecting blocks with almost equal variance, we thus choose the partition δ^* for which the difference in the variances of the two distributions is minimized, i.e.

$$\delta^* = \underset{\delta \in \{0,1\}^d}{\operatorname{argmin}} \left| Var\left(S_{\delta}^{\pi}\right) - Var\left(S_{1-\delta}^{\pi}\right) \right|.$$

Equivalently,

$$\delta^* = \underset{\delta \in \{0,1\}^d}{\operatorname{argmin}} \left| Cov\left(S_{\delta}^{\pi}, S^{\pi}\right) - Cov\left(S_{1-\delta}^{\pi}, S^{\pi}\right) \right|,$$

since $Var(S_{\delta}^{\pi}) - Var(S_{1-\delta}^{\pi}) = Cov(S_{\delta}^{\pi}, S_{\delta}^{\pi} + S_{1-\delta}^{\pi}) - Cov(S_{1-\delta}^{\pi}, S_{\delta}^{\pi} + S_{1-\delta}^{\pi})$. The latter formulation of the block selection problem is relatively easy to solve. In fact, denoting $\beta_j^{\pi} = Cov(X_j^{\pi}, S^{\pi})$, we can express $Cov(S_{\delta}^{\pi}, S^{\pi}) = \sum_{j=1}^{d} \delta_j \beta_j^{\pi}$. Finding two blocks with row sums that have equal variance is thus an example of the *number partitioning* problem, where the set $\{\beta_1^{\pi}, \beta_2^{\pi}, \ldots, \beta_d^{\pi}\}$ needs to be partitioned so that $\sum_{j=1}^{d} \delta_j \beta_j^{\pi} = \sum_{j=1}^{d} (1 - \delta_j) \beta_j^{\pi}$ for the corresponding δ. A simple and intuitive heuristic for this number partitioning problem is the so-called *greedy algorithm*, which iterates through the numbers and places the largest remaining number (in absolute value) into the subset that yields the smallest difference between the respective

[12] Recall that $E[S] = \frac{1}{n} \sum_{j=1}^{n} s_j$ is invariant under intra-column rearrangements.

sums.[13] Another possibility is to use the *KK differencing algorithm* (Karmarkar and Karp 1982), which is known to be very effective in finding optimal partitions, but is slower.

It is unlikely that the above block selection rule yields an optimal rearrangement after one iteration, and thus, in the following iterations one needs to ensure that the chosen blocks are sufficiently different from the blocks in the previous iteration. We achieve this in two ways. *Firstly*, starting from the second iteration, we apply the number partitioning algorithm to find the blocks with equal variance sequentially on the previously obtained blocks. More precisely, starting from the second iteration, we adjust the block selection rule by first partitioning $\{\beta_j^\pi : \delta_j = 1\}$ (where δ is the previous partition), obtaining a difference $\tilde{\beta}$ in variances. We then partition $\{\tilde{\beta}\} \cup \{\beta_j^\pi : \delta_j = 0\}$. Doing so induces a partitioning of the entire set $\{\beta_1^\pi, \beta_2^\pi \ldots, \beta_d^\pi\}$, which is typically different from the partition in the previous iteration. *Secondly*, in order to avoid that the algorithm gets stuck in a partition for which the blocks are already oppositely ordered (and the rearrangement does not reduce the variability measure), it is specified that when this happens, a randomly chosen block selection is used instead of the one based on variance equalization. The pseudo-code describing the sequence of steps for the Block RA with blocks selected to have similar variances is shown in Algorithm 2. Similarly to Algorithm 1, it requires setting a target value for the objective function a^* and the maximum number of iterations M.

Algorithm 2: Pseudo-code of the **BRAVE** algorithm to minimize the variability V of vector containing all row sums of a matrix X. Blocks are selected to have similar variances. If no improvement is observed, a random block is chosen

1. Set target value a^* and maximum number of iterations M;
2. Set π to a randomly chosen intra-column arrangement of the elements;
3. Set $a := V(S^\pi)$ with $S^\pi := \sum_{j=1}^d X_j^\pi$ and let $m = 0$;
4. **while** $a > a^*$ *and* $m < M$ **do**
   5. Block selection: Find $\delta^* = \text{argmin}_\delta \, |\text{Cov}(S_\delta^\pi, S^\pi) - \text{Cov}(S_{1-\delta}^\pi, S^\pi)|$;
   6. Rearrange the rows of the submatrix $X_{\delta^*}^\pi$ so that the row sums $S_{\delta^*}^\pi$ become oppositely ordered to $S_{1-\delta^*}^\pi$, and compute $a_{\delta^*} := V(S_{\delta^*}^\pi + S_{1-\delta^*}^\pi)$;
   7. Set $m = m + 1$;
   8. **if** a_{δ^*} *is equal to* a **then**
      9. Set δ to a randomly chosen partition;
      10. Rearrange the rows of the submatrix X_δ^π so that the row sums S_δ^π become oppositely ordered to $S_{1-\delta}^\pi$, and compute $a := V(S_\delta^\pi + S_{1-\delta}^\pi)$;
      11. Set $m = m + 1$;
   else
      12. Set $a = a_{\delta^*}$;
   end
end

[13] The greedy algorithm has a running time of $O(d \log d)$ and is flexible enough to deal with real numbers. Most partitioning algorithms are designed to work with integer numbers or positive real numbers. See Korf (1998) for a review of alternative solutions.

We also consider a modification of BRAVE, which consists in permanently switching from BRAVE to the standard RA after a series of initial iterations. We label this hybrid approach as BRAVE + RA and will further investigate it in Sect. 5.

4 BRAVE as generalization and improvement of number partitioning algorithms

In this section, we explore the connection between BRAVE and number partitioning algorithms. We first explain that any rearrangement algorithm (and thus also BRAVE) can be used to deal with the number partitioning problem. Next we show that BRAVE can actually be seen as a generalization and improvement of number partitioning algorithms.

Assume that a set of d real numbers with the total sum s is given, and that we would like to partition the set into two subsets with sums that are as similar as possible. This problem can be cast as a matrix rearrangement problem. Indeed, by adding d zeros to this set, we can represent the numbers as a matrix $X \in \mathbb{R}^{2\times d}$, in which the second row contains only zeros. Let S denote the corresponding vector of row sums. Clearly, number partitioning the set amounts to finding a rearranged matrix X^{π^*} such that S^{π^*} is as constant as possible (and in the ideal situation, only assumes the value $s/2$). It thus follows that rearrangement algorithms can be used to solve the number partitioning problem. Among these rearrangement algorithms, BRAVE has the particular feature that it uses a number partitioning algorithm in its set-up.

4.1 Generalization of number partitioning algorithms

Applying BRAVE to the matrix $X \in \mathbb{R}^{2\times d}$ that we just described indeed requires solving a number partitioning problem in each iteration, namely when finding the partition of the column covariances (see Sect. 3.2). In fact, in the first iteration, partitioning the covariances between the columns of X and the row sums of X is equivalent to partitioning the numbers in the first rows of X. Indeed, since the first row contains d numbers (denote them by $a_1, \ldots, a_d$) and the second row only zeros, it follows that in the first iteration of BRAVE the covariance of the jth column with the row sum vector $(s, 0)^\top$ is equal to $s \cdot a_j/2 - s/2 \cdot a_j/2 = s \cdot a_j/4$. Hence, the covariances that we need to be partition are proportional to the given numbers. Therefore, after one iteration, the non-zero elements of block 1 and block 2 yield the same subsets as one would have obtained by applying the number partitioning algorithm to the given set of d numbers. In other words, when applied to the given $X \in \mathbb{R}^{2\times d}$, BRAVE yields the same solution as the one provided by the number partitioning algorithm. However, BRAVE can be applied to general $n \times d$ matrices and thus generalizes number partitioning algorithms in this regard.

4.2 Improvement of number partitioning algorithms

Whilst application of BRAVE to the number partitioning problem yields after one iteration the same partition as given by the same partitioning algorithm, BRAVE

continues to iteratively select new blocks and to rearrange them, which can only bring the solution closer to optimality. In other words, since the matrix corresponding to the solution after one iteration may not be Σ-countermonotonic, BRAVE can never perform worse than the algorithm used to partition the covariances and may strictly outperform it.

In what follows, we focus on the KK differencing algorithm and the greedy algorithm. When BRAVE is implemented with the KK or greedy algorithm to find blocks with equal variance, we henceforth denote this by the BRAVE(KK) and BRAVE(greedy) algorithm, respectively.

Example 1 [Comparing KK with BRAVE(KK)] Hayes (2002) provides two relevant test cases for the analysis of the performance of a partitioning algorithm. The first test case is shown in Figure 1 of Hayes (2002), in which he illustrates the case of partitioning the set $T = \{484, 114, 205, 288, 506, 503, 201, 127, 410\}$ into two subsets T_1 and T_2 such that their total sums t_1, respectively, t_2 are as close as possible. Both the greedy and the differencing algorithm of KK do not succeed and yield that $|t_1 - t_2|$ is equal to 52 and 28, respectively. By contrast, the BRAVE(KK) needs on average around 17 iterations to find the (unique) perfect partition of dividing the numbers into $T_1 = \{484, 114, 205, 288, 201, 127\}$ and $T_2 = \{506, 503, 410\}$. The second test case studied by Hayes (2002) is the problem of partitioning the set $T = \{771, 121, 281, 854, 885, 734, 486, 1003, 83, 62\}$ with total sum $s = 5280$. Applying the differencing algorithm of KK yields $|t_1 - t_2| = 26$. However, applying the BRAVE(KK) reveals that a perfect partition can be found using only two variance equalization steps. First, the set is split into $\{121, 281, 854, 885, 486\}$ and $\{771, 734, 1003, 83, 62\}$ with total sums 2627 and 2653. Then, the two sets are rearranged into $T_1 = \{771, 281, 854, 734\}$ and $T_2 = \{121, 885, 486, 1003, 83, 62\}$, with $t_1 = t_2 = 2640$. Note also that applying the greedy algorithm would yield a discrepancy of 32 between the two subset sums, and also that application of the classic RA yields the same solution as the Greedy algorithm (this also holds for the first test-case).[14] The second test-case of Hayes (2002) is the result of drawing $d = 10$ integer numbers between 1 and 2^d. Hereafter, we consider number partitioning when larger values of d are used.

In order to compare the performance of the various algorithms that are applicable to the number partitioning problem, we take into account the observations in Gent and Walsh (1998) and Mertens (1998); namely, that the size d of the set to be partitioned, and the possible range of the numbers, affect the difficulty of the optimization problem in a specific way. In particular, instances with d integers drawn uniformly from $\{1, \ldots, 2^d\}$ are typically difficult to solve, and only one partition, on average, is *perfect* (with the difference in sums $\Delta := |t_1 - t_2| \leq 1$). We therefore generate $N = 1000$ random instances as described above, for each $d = 10, 11, \ldots, 20$. We choose these relatively low values for d, as this allows solving the problem exactly, using a dynamic programming formulation of this problem; see e.g. Gent and Walsh

[14] In fact, from Proposition A.1 in Bernard et al. (2017) it follows that the greedy algorithm yields an arrangement in which each column is oppositely ordered to the sum of the other columns and is thus—in the particular context of the partition problem—as good as the RA.

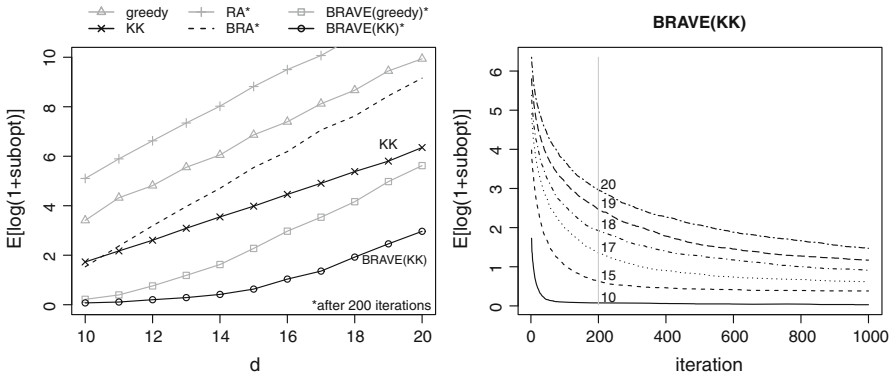

Fig. 1 *Left panel* suboptimality of the greedy and KK solutions, as well as the solutions of the rearrangement algorithms after 200 iterations. By BRAVE(greedy) and BRAVE(KK) we refer to the two versions of the BRAVE algorithm, which use the greedy, respectively, the KK algorithm to partition the column covariances (see Sect. 3.2). *Right panel* the development of average suboptimality of BRAVE(KK) solution, as a function of the number of iterations (rearrangements). The *lines* correspond to selected values of $d \in \{10, \ldots, 20\}$

(1998). Note, however, that the heuristic algorithms can also be applied to much larger problem instances. The corresponding average (log-)suboptimality is computed as

$$\frac{1}{N}\sum_{i=1}^{N} \log\left(1 + \Delta_i^{\text{Heur}} - \Delta_i^{\text{Opt}}\right),$$

where Δ_i^{Heur} and Δ_i^{Opt} are the differences given by the heuristic, respectively, by the optimal solution, for the ith problem instance. The adjustment by 1 inside the logarithm is to count as $\log(1) = 0$ the cases when a heuristic attains the optimal solution.

In Fig. 1, left panel, the average suboptimality is plotted for the various heuristics, depending on d. For the rearrangement algorithms, 200 iterations are used. The poorest performer is the RA, followed by the greedy algorithm and the (random) BRA. The best results are given by the two versions of the Block RA that use variance equalization when selecting the blocks; they have a lower suboptimality than KK. BRAVE(KK) is clearly the best performer. Note that the partition obtained after one iteration is the same as obtained when applying KK applied to the original set of numbers. Further iterations look for a partition of the matrix into two blocks with row sums that are not oppositely ordered and will bring the solution closer to optimality. To illustrate this convergence, the right panel of Fig. 1 shows the suboptimality of BRAVE(KK) for different sizes d of the set to be partitioned, as a function of the applied rearrangements (iterations).

Block rearrangement algorithms are useful heuristics to solve not only the number partition problem, but also for solving the n-partitioning problem, as we explain in Remark 2 below.

Remark 2 (The n-partitioning problem) The so-called n-partitioning problem is the problem of partitioning a given set of numbers into n sets with sums that are as equal as possible (Chopra and Rao 1993). Clearly, block rearrangement algorithms can be used as heuristics to solve the n-partitioning problem. Indeed, assume that a set of d real

numbers with the total sum s is given. We add $(n-1)d$ zeros to this set, so that the total cardinality is nd, and we can represent the numbers as a matrix $X \in \mathbb{R}^{n \times d}$, where the last $n-1$ rows contain only zeros. Let S denote the corresponding vector or row sums. Partitioning the set into n subsets with sums that are as similar as possible, amounts to finding a rearranged matrix X^{π^*} such that S^{π^*} is as constant as possible (and in the ideal situation, only assumes the value s/n). The n-partitioning problem is also closely related to the parallel machine scheduling problem. Given d jobs with processing times $p_1, \ldots, p_d$ and n parallel identical machines which can execute at most one job at a time, one aims at assigning each job to exactly one machine, so as to minimize the latest completion time of all jobs. This problem can be cast as a n-partitioning problem and rearrangement algorithms can be applied to this problem as competitors to other known heuristics proposed in Frangioni et al. (2004) and Alvim and Ribeiro (2004). Exact algorithms are presented in Dell'Amico and Martello (1995) (branch-and-bound algorithm), Mokotoff (2004) (cutting-plane algorithm) and Dell'Amico et al. (2008) (algorithm based on a specialized binary search and a branch-and-price scheme). Dell'Amico and Martello (2005) show that, on the same types of instances, the algorithm of Dell'Amico and Martello (1995) is faster than the one of Mokotoff (2004) by some orders of magnitude.

5 Performance evaluation of the block rearrangement algorithms

All rearrangement algorithms described in the previous sections are designed to improve the objective function (satisfying Assumption 1) at each iteration. They differ in terms of the selected (block of) column(s) that is rearranged at each iteration and thus in terms of the magnitude of the improvement. In this section, we compare the effectiveness of the RA and various versions of the Block RA for minimizing the variance of the row sums, by rearranging matrices that have $n = 100$ or $n = 1000$ rows and $d = 100$, 1000, or 10,000 columns. Specifically, we compare RA, BRA and BRAVE and also consider the BRAVE+RA procedure, which aims at combining the best of the two worlds by using BRAVE for the initial rearrangements and then permanently switching to RA. Ideally, the BRAVE+RA procedure switches whenever the future rearrangements using BRAVE are less effective than when RA is used. In practice, there is no simple method available to predict the difference in performance using BRAVE and RA in the subsequent iterations. Based on a comparison of several plausible approaches on different data sets, we recommend to use BRAVE for the first 30 iterations or until no improvement is observed, and then proceed further with RA. Doing so, BRAVE is used as a pre-solver for RA. The upper bound of 30 on the number of BRAVE iterations is to be considered as a rule-of-thumb, which, in some applications may be too low, in others too high, but, from our experience, a reasonable general-purpose value to limit the number of BRAVE iterations.[15]

[15] The results for the BRAVE algorithms are based on the greedy algorithm for finding the blocks. Using the KK algorithm, we obtained sightly better results, but the computation times were longer. We also tested the Block RA1 algorithm of Bernard and McLeish (2016) with block selection as in (2). These simulations showed that block selection based on maximal Spearman correlation of the row sums in a few iterations leads to a rearranged matrix with maximal Spearman correlation that is close to -1, but the improvement

We expect that the first few iterations in the block rearrangement algorithms will have the largest (absolute) impact on the objective functions and that afterwards the improvements will become relatively small. We further expect that these algorithms (in particular, BRAVE) will yield greater initial improvements than the RA and that the gains will tend to be more pronounced when the number of columns d is large and the matrix columns are dissimilar. We verify this intuition by sampling and rearranging $N = 100$ matrices for 12 possible configurations. Besides varying $n \in \{100, 1000\}$ and $d \in \{10^2, 10^3, 10^4\}$, we also vary the choice of the distribution used to generate the matrices. The matrix elements are random draws from either the uniform distribution on [0, 1], the Pareto distribution with shape parameter equal to 2 and a unit scale parameter, or a random variable generated as the product between a Beta random variable with shape and scale equal to 0.5 and a Bernoulli random variable that takes the value of one with probability $1/n$, and is zero otherwise. These three distributional choices represent different scenarios. In the case of the uniform distribution, all columns are similar. By contrast, in the Pareto case, even though all columns are generated using draws from the same distribution, the columns can be very dissimilar due to the fat tail of this distribution. Finally, the matrices obtained using the Bernoulli–Beta set-up are similar to those used in the case of n-partitioning, since on average there are $n - 1$ zeros per column. We have tested several other distributional choices (such as the exponential distribution and the Beta distribution), but our findings can be summarized using these three cases.

In Figs. 2, 3 and 4, we plot the progress of each algorithm in terms of reducing the variance of the row sums. In particular, we sampled $N = 100$ different matrices with entries from the uniform, Pareto or Bernoulli–Beta distribution, respectively, and plotted the average value of the natural logarithm of the variance of S^π after $i = 1, \ldots, 500$ iterations. Namely, the plots show

$$\frac{1}{N}\sum_{j=1}^{N} \log\left(Var\left(\left(S^\pi\right)_{i,j} \right) \right),$$

where $(S^\pi)_{i,j}$ is the vector of row sums after the ith iteration of the jth experiment run. The solid gray line corresponds to the trajectory of the RA, the black dotted line is the average evolution of the objective when random BRA is used, and the gray and black dashed lines correspond to rearrangement using BRAVE and BRAVE + RA.

Consider first the trajectories in Fig. 2, which shows the results from rearranging matrices with entries sampled from the standard uniform distribution. As expected, we observe that for the initial rearrangements, in all cases considered, larger improvements in the objective function are obtained by rearranging blocks rather than single columns. However, the RA eventually catches up in each setup and eventually dominates BRA

Footnote 15 continued
in the objective function V is relatively low compared to the RA, BRAVE and BRAVE + RA algorithms. Moreover, the Spearman correlation approach requires sorting the partial row sums for all considered partitions at every iteration, and is time-consuming. For the sake of brevity, we do not report these results.

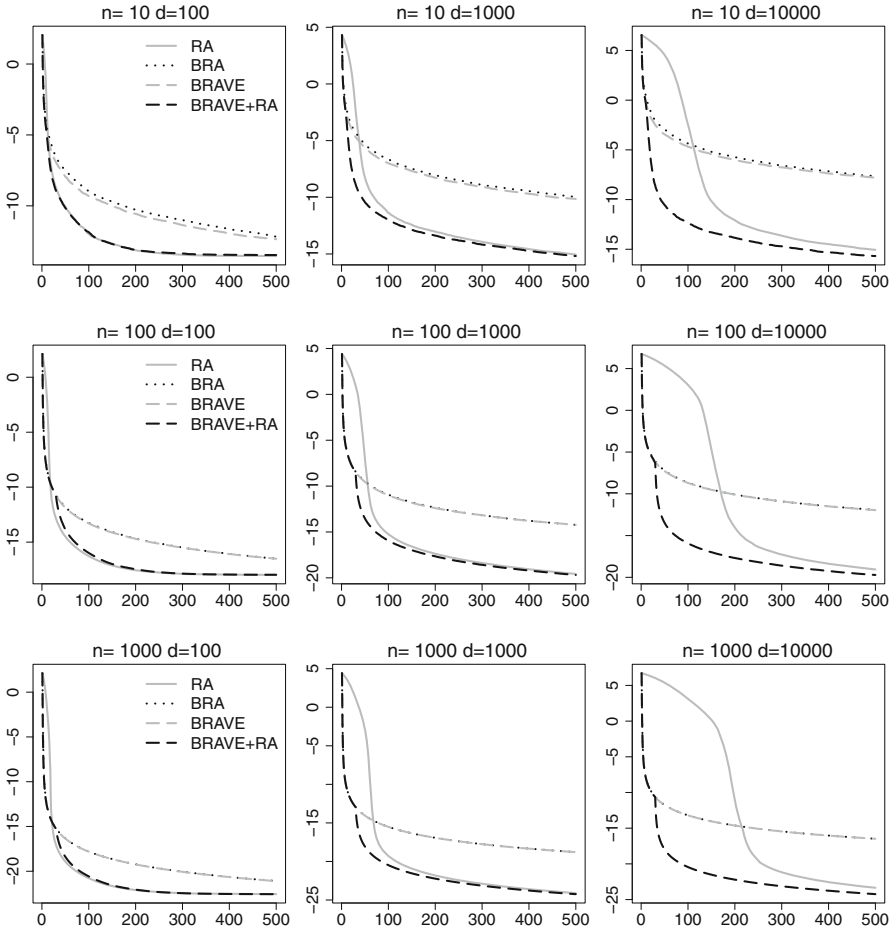

Fig. 2 The effect of the rearrangement algorithms on the trajectory of the natural logarithm of the variance of the row sums as a function of the number of iterations (rearrangements), when the matrix entries are sampled independently from a standard uniform distribution. The *lines* show the averages over 100 replications for matrices with the given number or rows n and number of columns d

and BRAVE, which have similar performance.[16] The best results are obtained by BRAVE + RA (black dashed line) which outperforms all other algorithms with RA a close second best. Overall, it appears that in the case of matrices with similar columns, the classical RA catches up quickly with the block rearrangement algorithms, provided a moderately high number of iterations is used. Using BRAVE + RA provides a further modes improvement and indicates that BRAVE is a valuable pre-solver to deal with rearrangement problems.

Figure 3 shows the results when the entries of the matrix are sampled from the fat-tailed Pareto distribution. Because of the heavy tails, some columns contain out-

[16] This is to be expected. The columns in this experiment are similar and on average BRA chooses blocks of equal size so that the variances of their row are typically close to each other.

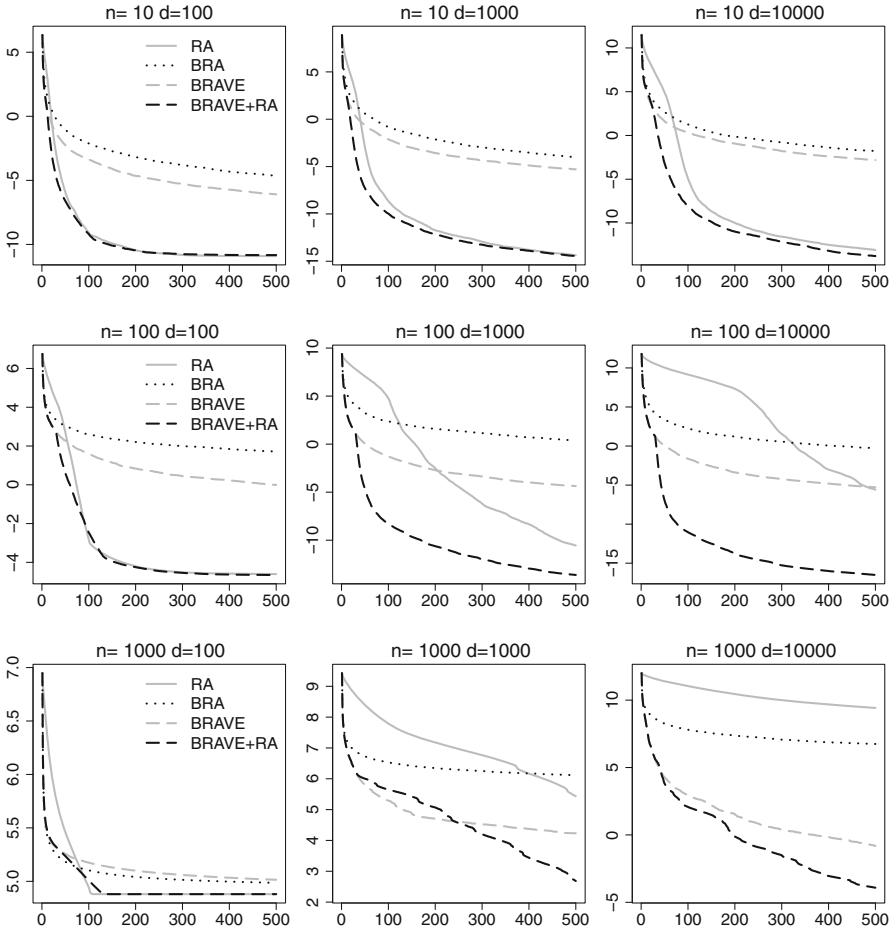

Fig. 3 The effect of the rearrangement algorithms on the trajectory of the natural logarithm of the variance of the row sums as a function of the number of iterations (rearrangements), when the matrix entries are sampled independently from a Pareto distribution with shape parameter 2 and unit scale. The *lines* show the averages over 100 replications for matrices with the given number or rows n and number of columns d

lying values and one can expect that the selection method for the block of columns to be rearranged matters. Indeed, the proposed BRAVE heuristic, where blocks are selected based on variance equalization, clearly dominates the random block selection approach. Furthermore, BRAVE+RA significantly outperforms the RA (solid gray line) in terms of finding a lower objective function within a given number of iterations, for $d \geq 1000$.

Another extreme case of heterogeneity is presented in Fig. 4. Here, each element is either zero, or, with a probability of $1/n$ equal to a draw from a Beta distribution with shape and scale parameters equal to 0.5. In this case, the matrix consists predominantly of zeros and the RA loses a lot of its effectiveness, since only the non-zero elements in the column are rearranged per iteration. When n is larger than d, the optimal configuration is on average equal to the matrix with one non-zero element per row. It follows

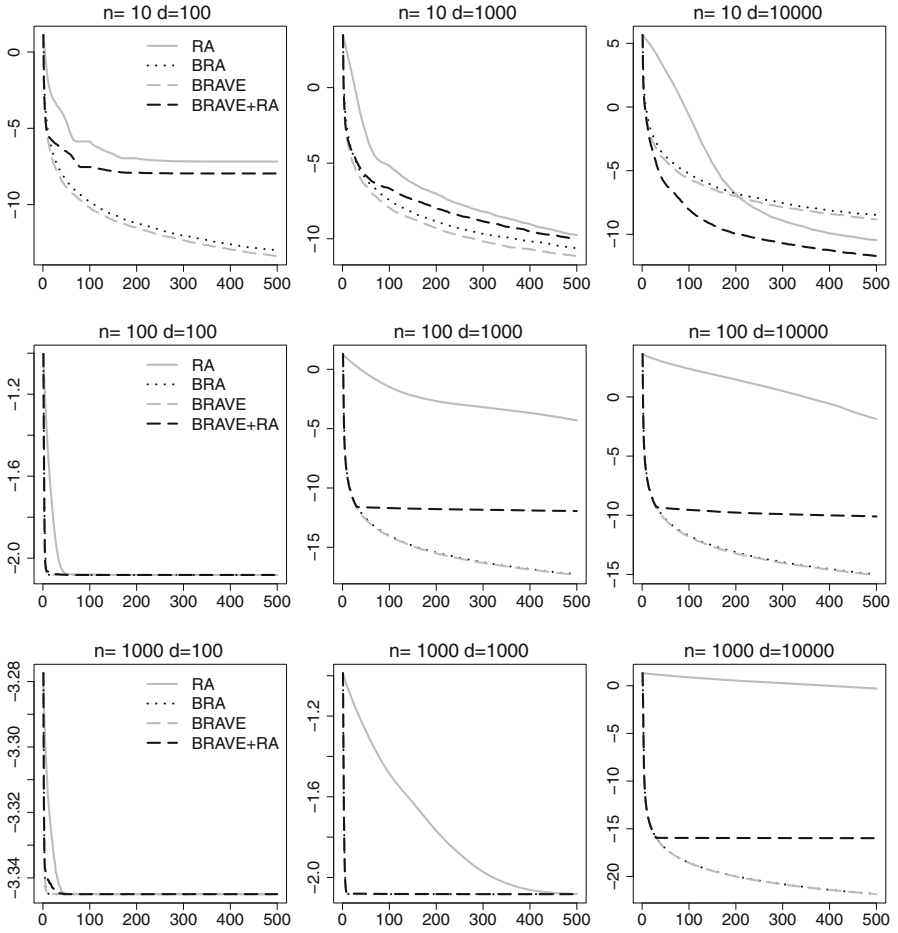

Fig. 4 The effect of the rearrangement algorithms on the trajectory of the natural logarithm of the variance of the row sums as a function of the number of iterations (rearrangements), when the matrix entries are independent realizations of the product between a Beta distribution with shape and scale equal to 0.5, and a Bernoulli random variable that is one with a probability of $1/n$, and zero otherwise. The *lines* show the averages over 100 replications for matrices with the given number or rows n and number of columns d

that in the cases $n = 100, d = 100$, $n = 1000, d = 100$ and $n = 1000, d = 1000$, typically only one block rearrangement is needed to rearrange in such a way that each row has one non-zero element. The RA needs several iterations to converge to that situation. The more interesting cases are when d is large compared to n. In these instances, we observe that the RA algorithm is unable to reach the same low variance as BRA, BRAVE and BRAVE+RA within 500 iterations.

The different numerical experiments confirm the intuition that the block rearrangement algorithms achieve larger initial improvements in the objective function than the classical RA. When the columns are similar (homogeneous), the classical RA catches up with block rearrangement algorithms and even outperforms them. BRAVE+RA outperforms RA, but the improvement is small. When the matrix has columns that

Table 1 The average running times in seconds (across 100 replications) for the rearrangement algorithms used to minimize the variance of the row sums of $n \times d$ matrices, filled with random draws from a uniform distribution

Iter	$n = 10$			$n = 100$			$n = 1000$		
	$d = 100$	$d = 1000$	$d = 10{,}000$	$d = 100$	$d = 1000$	$d = 10{,}000$	$d = 100$	$d = 1000$	$d = 10{,}000$
RA									
50	0.01	0.02	0.11	0.01	0.11	1.11	0.10	1.39	13.30
100	0.02	0.03	0.21	0.03	0.20	2.22	0.20	2.80	26.85
300	0.05	0.09	0.61	0.09	0.57	6.67	0.58	8.42	81.07
500	0.09	0.15	1.02	0.16	0.93	11.12	0.97	14.04	135.28
BRA									
50	0.01	0.02	0.17	0.02	0.15	1.92	0.12	2.28	20.43
100	0.02	0.05	0.35	0.04	0.31	3.90	0.25	4.59	40.74
300	0.07	0.14	1.04	0.13	0.94	11.84	0.76	13.91	121.67
500	0.12	0.23	1.74	0.21	1.56	19.81	1.26	23.23	202.48
BRAVE									
50	0.03	0.15	1.45	0.05	0.41	4.52	0.22	3.17	28.92
100	0.06	0.28	2.66	0.10	0.79	8.88	0.44	6.41	57.97
300	0.17	0.78	7.33	0.28	2.08	24.17	1.32	19.36	173.84
500	0.28	1.27	11.91	0.45	3.29	38.40	2.19	32.19	288.71
BRAVE + RA									
50	0.01	0.05	0.41	0.04	0.29	3.23	0.17	2.51	23.02
100	0.02	0.06	0.51	0.05	0.38	4.58	0.27	4.14	37.10
300	0.06	0.12	0.93	0.12	0.77	9.98	0.66	10.66	93.11
500	0.09	0.18	1.34	0.18	1.16	15.39	1.04	17.18	149.11

The reported times are obtained using R-3.3.1 on a 3.2 GHz Intel Xeon processor and 6 GB of RAM, running on Windows 7

are highly heterogeneous, the method of choosing the blocks matters. The BRAVE approach consistently outperforms the BRA, and BRAVE + RA improves significantly on the classical RA. As a consequence, BRAVE appears to be a useful pre-solver for the RA. An important caveat is that when the matrix is predominantly filled with zero elements, rearrangement of blocks of columns should always be preferred over the traditional RA; see also the study of the number partitioning problem in Sect. 4. The gains in performance of block rearrangements come at the price of higher running times. We show this in Table 1 for the case of rearranging matrices with elements from the standard uniform distribution.[17] The complexity of each iteration of the RA and the BRA is $O(nd + n\log(n))$, where the nd term is due to computing row-sums, and $n\log(n)$ is due to sorting the obtained column vectors. For BRAVE, the complexity is

[17] To save space we do not report the running times for the Pareto and the Bernoulli–Beta set-ups, since the obtained results for the running times are qualitatively similar. This is expected, since the nature of the randomized matrix entries has little impact on the computer time spent on the sorting and summation operations done in the rearrangement algorithms.

$O(nd + n\log(n) + d\log(d))$, where the additional last term is due to variance equalization by the greedy algorithm, which sorts the d covariances between each column and the total row-sum vector. Note that computing these covariances is also included in the nd term. From Table 1 we can observe that the increase in computation time for all algorithms is approximately proportional to both n and d, and the logarithmic terms contribute little. We further see that, in all cases, the RA is the fastest, but the BRAVE+RA is a close second best. Comparing random block selection (BRA) with the block selection using variance equalization, we find that the additional cost in relative terms becomes smaller as the number of columns increases.

6 Conclusion

In this paper, we discussed the problem of finding the intra-column rearrangement of a matrix that minimizes the variability among its row sums. We discuss characteristics of the optimal solution to this challenging combinatorial problem and we present methodological innovations relative to the current literature. In particular, as an alternative to rearranging single columns (RA) or randomly chosen blocks of columns (BRA), we propose the BRAVE algorithm, which rearranges blocks that are as similar as possible (in terms of the variance of their row sums). For the problem of 2-partitioning (number partitioning), the BRAVE algorithm appears superior to well-known existing number partitioning algorithms such as the KK differencing algorithm. When large dense (homogenous) matrices need to be rearranged, we provide numerical evidence that while block rearrangement algorithms (in particular BRAVE) yield a sharp initial improvement in the objective function, the RA catches up quickly and the improvement of using BRAVE as a pre-solver for RA (BRAVE+RA approach) is modest. For large heterogeneous matrices, however, the RA is outperformed by BRAVE and BRAVE + RA.

Compliance with ethical standards

Conflict of interest The authors declare that there are no conflicts of interest.

References

Alvim AC and Ribeiro CC (2004) A hybrid bin-packing heuristic to multiprocessor scheduling. In: International workshop on experimental and efficient algorithms. Springer, Berlin, pp 1–13

Bernard C, McLeish D (2016) Algorithms for finding copulas minimizing convex functions of sums. Asia Pac J Oper Res **33**(5):1650040. doi:10.1142/S0217595916500408

Bernard C, Rüschendorf L, Vanduffel S (2015) Value-at-risk bounds with variance constraints. J Risk Insur. doi:10.1111/jori.12108

Bernard C, Rüschendorf L, Vanduffel S, Yao J (2017) How robust is the value-at-risk of credit risk portfolios? Eur J Financ 23(6):507–534

Boland PJ, Proschan F (1988) Multivariate arrangement increasing functions with applications in probability and statistics. J Multivar Anal 25(2):286–298

Chopra S, Rao MR (1993) The partition problem. Math Program 59(1–3):87–115

Coffman E, Yannakakis M (1984) Permuting elements within columns of a matrix in order to minimize maximum row sum. Math Oper Res 9(3):384–390

Day PW (1972) Rearrangement inequalities. Can J Math 24(5):930–943

Dell'Amico M, Martello S (1995) Optimal scheduling of tasks on identical parallel processors. ORSA J Comput 7(2):191–200
Dell'Amico M, Martello S (2005) A note on exact algorithms for the identical parallel machine scheduling problem. Eur J Oper Res 160(2):576–578
Dell'Amico M, Iori M, Martello S, Monaci M (2008) Heuristic and exact algorithms for the identical parallel machine scheduling problem. INFORMS J Comput 20(3):333–344
Embrechts P, Puccetti G, Rüschendorf L (2013) Model uncertainty and VaR aggregation. J Bank Financ 37(8):2750–2764
Frangioni A, Necciari E, Scutella MG (2004) A multi-exchange neighborhood for minimum makespan parallel machine scheduling problems. J Comb Optim 8(2):195–220
Gent IP, Walsh T (1998) Analysis of heuristics for number partitioning. Comput Intell 14(3):430–451
Graham RL (1966) Bounds for certain multiprocessing anomalies. Bell Syst Tech J 45(9):1563–1581
Hayes B (2002) Computing science: the easiest hard problem. Am Sci 90(2):113–117
Hsu W-L (1984) Approximation algorithms for the assembly line crew scheduling problem. Math Oper Res 9(3):376–383
Jakobsons E, Wang R (2016) Negative dependence in matrix arrangement problems. http://ssrn.com/abstract=2756934
Karmarkar N, Karp RM (1982) The differencing method of set partitioning. Technical Report UCB/CSD 82/113, Computer Science Division, University of California, Berkeley
Korf RE (1998) A complete anytime algorithm for number partitioning. Artif Intell 106(2):181–203
Marshall AW, Olkin I, Arnold BC (2011) Inequalities: theory of majorization and its applications, 2nd edn. Springer, New York
Mertens S (1998) Phase transition in the number partitioning problem. Phys Rev Lett 81(20):4281
Mokotoff E (2004) An exact algorithm for the identical parallel machine scheduling problem. Eur J Oper Res 152(3):758–769
Puccetti G, Rüschendorf L (2012) Computation of sharp bounds on the distribution of a function of dependent risks. J Comput Appl Math 236(7):1833–1840
Puccetti G, Wang R (2015) Extremal dependence concepts. Stat Sci 30(4):485–517
Rüschendorf L (2013) Mathematical risk analysis. In: Mikosch TV, Resnick SI, Robinson SM (eds) Springer Series in Operations Research and Financial Engineering. Springer, Heidelberg
Shaked M, Shanthikumar JG (2007) Stochastic orders. Springer, Berlin
Wang B, Wang R (2011) The complete mixability and convex minimization problems with monotone marginal densities. J Multivar Anal 102(10):1344–1360
Wang B, Wang R (2016) Joint mixability. Math Oper Res 41(3):808–826