On some synchronization problems with multiple instances

Dries Cornilly,* Giovanni Puccetti,† Ludger Rüschendorf,‡ and Steven Vanduffel.§

March 25, 2021

Abstract

Many classical synchronization problems such as the assembly line crew scheduling problem (ALCS), some data association problems or multisensor tracking problems can be formulated as finding intra-column rearrangements for a single matrix representing costs, distances, similarities or time requirements. In this paper, we consider an extension of these problems to the case of *multiple* matrices, reflecting various possible instances (scenarios). To approximate optimal rearrangements, we introduce the Block Swapping Algorithm (BSA) and a further customization of it that we call the customized Block Swapping Algorithm (Cust BSA). A numerical study shows that the two algorithms we propose – in particular Cust BSA – yield high-quality solutions and also deal efficiently with high-dimensional set-ups.

Key-words: Assembly line crew scheduling (ALCS); Swapping algorithm; Rearrangement algorithm; Multidimensional assignment problems.

*Dries Cornilly (email: dcornilly@asteria-im.com). KULeuven and Asteria Investment Managers SA, Rue du Rhône 62, 1204 Genève, Switzerland.

†Giovanni Puccetti (email: giovanni.puccetti@unimi.it). University of Milan, Via Conservatorio 7, 20122 Milano, Italy.

‡Ludger Rueschendorf (e-mail: ruschen@stochastik.uni-freiburg.de). University of Freiburg, Ernst-Zermelo-Straße 1, 79104 Freiburg, Germany

§Steven Vanduffel, corresponding author (e-mail: steven.vanduffel@vub.ac.be). Vrije Universiteit Brussel, Pleinlaan 2, 1050, Brussels, Belgium.

1 Introduction

The synchronization problems with multiple instances that we consider in this paper can be seen as extensions of some multidimensional assignment problems. These problems are well studied in the literature and have various applications such as the planning of workforces in production systems with multiple synchronous workstations, the design of multitarget multisensor tracking systems, data association problems, and computer vision. For general treatments of multidimensional assignment problems and their solution in particular cases, we refer to Haley (1963), Pierskalla (1968), Rüschendorf (1983), Gilbert and Hofstra (1988), Bar-Shalom (1990), Poore, Rijavec, Liggins, and Vannicola (1993), Pardalos and Pitsoulis (2000), Spieksma (2000), Karapetyan and Gutin (2011), and the many references therein. The classical Assembly Line Crew Scheduling problem (ALCS) with one problem instance has been studied in Coffman Jr and Yannakakis (1984) and in Hsu (1984), while the related workforce planning problem is investigated in Lee and Vairaktarakis (1997), Vairaktarakis, Cai, and Lee (2002), and in Camm, Magazine, Polak, and Zaric (2008).

We formulate extensions of these synchronization problems in the context of ALCS. In ALCS it is assumed that the production of some item requires the serial execution of d jobs and that there are n identical assembly lines available. Assume further that for the j-th job ($j = 1, \ldots, d$) there is a pool of n specialized workers available, such that the i-th worker ($i = 1, \ldots, n$) requires a deterministic labour time x_{ij} to complete the j-th job, i.e., there is one problem instance. The time needed for the i-th assembly line to produce one good is then given by the sum $s_i = \sum_{j=1}^{d} x_{ij}$. The objective of the classical ALCS is to arrange the workers on the n assembly lines so that the production times or labour costs over all lines become as synchronized as possible. If one represents the labour costs by the $n \times d$ cost matrix $\mathbf{X} = (x_{ij})$, the problem can be formulated as permuting the elements within each column of $\mathbf{X}$ such that the maximum of the row sums s_i of the rearranged matrix is minimum; see Coffman Jr and Yannakakis (1984). Alternatively, one may also aim to minimize the variance among the row sums (Pesko (2006), Pesko and Hajtmanek (2014)) in which case the problem is within the class of quadratic assignment problems (Burkard (1979), Burkard, Çela, Pardalos, and Pitsoulis (1998), Pardalos and Pitsoulis (2000), Loiola, de Abreu, Boaventura-Netto, Hahn, and Querido (2007)). A more general aim here is to characterize minimal elements of the vector $s := (s_i)$ of row sums with respect to the Schur order (see Rüschendorf (1983)), which implies the optimization of Schur-convex functions $f(s_1, \ldots\ldots, s_n)$ of the vector of row sums s_i. In this paper, we restrict ourselves to the variance formulation.

While for $d = 2$, the ALCS has a straightforward solution (arrange elements in the two columns in opposite order), it is well known that for $d > 2$ it is NP-hard. A successful heuristic approach, which has raised considerable interest in some specific applications in finance and actuarial science, is the Rearrangement Algorithm (RA), introduced in Puccetti and Rüschendorf (2012) and in Embrechts, Puccetti, and Rüschendorf (2013). The RA iteratively rearranges elements within the columns of the cost matrix $\mathbf{X}$ such that they become oppositely ordered to the row sums taken

across all other columns[1]. This heuristic relies on the fact that the property of opposite ordering to the sum is a necessary condition for optimality.

In various applications, however, the labour times of workers may not be known with certainty but are scenario dependent, i.e., we deal with multiple problem instances. Therefore, we assume that an initial assignment of the workers under the different problem instances can be represented by m different matrices $\mathbf{X}^k := (x_{ij}^k) \in \mathbb{R}^{n\times d}$, $k = 1, \ldots, m$, in which x_{ij}^k represents the labour time of the i-th worker allocated to the j-th job under the k-th *problem instance* (scenario). The problem is then to assign the workers to the different working lines such that under *all* possible scenarios the production times of the different production lines are as synchronized as possible, i.e., we need to rearrange the m matrices by the same set of column permutations such that their row sums exibit minimum variability. The assignment of the workers cannot be adapted to the specific scenario, but has to be chosen upfront without knowledge of the scenario at hand. This set-up is also applicable when dealing with the allocation of workers in the case they have to conduct a sequence of tasks (at the same machine).

In this paper, we do not embed the synchronization problem with multiple instances into the corresponding linear continuous assignment problem. Indeed, it is well-known that the convex polytope of such an assignment problem is not unimodular and allows for a large set of extreme points with no integral values, which leads to interpretation problems and to only rough bounds for the solutions. Instead, we propose the so-called *Block Swapping Algorithm* (BSA) where we use the variance as a criterion for measuring synchronization. Note however that this algorithm is also able to deal with more general synchronization criteria, as described above.

After formulating in Section 1.1 the synchronization problem as a (columns) rearrangement (assignment) problem of multiple matrices, we describe in Section 2 the BSA and a further customization of it that we call Cust BSA. Specifically, the customization consists in obtaining a suitable initial assignment after which one applies BSA. To obtain the initial assignment, a linear regression is applied to the initial matrices and next a generalized version of the so-called MinCov algorithm, developed by Cornilly, Puccetti, Rüschendorf, and Vanduffel (2020) for dealing with fair allocation problems, is used. To the best of our knowledge, there are no other algorithms available that deal with the extended ALCS problem we consider. In Section 3, we study the quality of the solutions obtained by BSA and Cust BSA. We find that, similar to the RA algorithm in the case of one problem instance, the BSA and its customized version are able to deal efficiently with the synchronization problem under a large number of problem instances.

The numerical experiments presented in this paper can be replicated by downloading the code of our algorithms at the web address https://github.com/cdries/SPMI.

[1]The RA turns out to be very useful in operations research (Boudt, Jakobsons, and Vanduffel (2018)) and in insurance where it appears as a reference tool for assessing the uncertainty with respect to risk estimates of portfolios (Bernard, Rüschendorf, Vanduffel, and Wang (2017), Hofert, Memartoluie, Saunders, and Wirjanto (2017)).

1.1 Formulation of synchronization problems as a (column) rearrangement problem for multiple matrices

We consider m matrices $\mathbf{X}^k = (x_{ij}^k) \in \mathbb{R}^{n\times d}$,

$$\mathbf{X}^k = \begin{bmatrix} x_{11}^k & x_{12}^k & \dots & x_{1d}^k \\ x_{21}^k & x_{22}^k & \dots & x_{2d}^k \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1}^k & x_{n2}^k & \dots & x_{nd}^k \end{bmatrix},\ k = 1, \dots, m.$$

The matrices $\mathbf{X}^k$, $k = 1, \dots, d$, represent problem instances. Let $\mathcal{A}$ be the set of all vectors $\pi = (\pi_1, \dots, \pi_d)$ in which the components π_j are permutations of $\{1, \dots, n\}$. Given a vector of permutations $\pi \in \mathcal{A}$, define $\mathbf{X}^{k,\pi} = (x_{ij}^{k,\pi})$ by

$$x_{ij}^{k,\pi} = x_{\pi_j(i)j}^k, \tag{1}$$

i.e., each π_j permutes the elements of the j-th column of *all* the matrices $\mathbf{X}^k$ to obtain the column permuted matrices $\mathbf{X}^{k,\pi}, k = 1, \dots, m$. In what follows, we call any $\pi \in \mathcal{A}$ a *rearrangement* and each matrix $\mathbf{X}^{k,\pi}$ a *rearranged matrix*. The rearranged matrix $\mathbf{X}^{k,\pi}$ represents the cost matrix of the classical ALCS under scenario k if one allocates workers via the rearrangement π.

Denote by $s_i^{k,\pi} = \sum_{j=1}^d x_{ij}^{k,\pi}, i = 1, \dots, n$, the row sums of $\mathbf{X}^{k,\pi}$, $k = 1, \dots, m$, and by $\overline{s}^k$ their mean value, i.e.,

$$\overline{s}^k = \frac{\sum_{i=1}^n s_i^{k,\pi}}{n} = \frac{\sum_{i=1}^n \sum_{j=1}^d x_{ij}^{k,\pi}}{n} = \frac{\sum_{i=1}^n \sum_{j=1}^d x_{ij}^k}{n}.$$

Notice that $\overline{s}^k$ does not depend on the specific rearrangement π. The variation of the production costs (time lenghts) under the k-th scenario is given by

$$V_k(\pi) = \frac{\sum_{i=1}^n (s_i^{k,\pi} - \overline{s}^k)^2}{n}.$$

Considering all m scenarios, the aim of the paper is to solve the optimal synchronization problem

$$V^* = \min_{\pi\in\mathcal{A}} V(\pi), \tag{2}$$

where $V(\pi)$ is the variance function defined as

$$V(\pi) = \frac{\sum_{k=1}^m V_k(\pi)}{m}. \tag{3}$$

Since the set $\mathcal{A}$ of admissible rearrangements is finite, there exists a solution to the optimal synchronization problem (2), i.e., a *minimum variance rearrangement* π^* rearranging all m matrices $\mathbf{X}^{k,\pi}, k = 1, \dots, m$, such that $V(\pi^*) = V^*$. Note that the optimal synchronization problem (2) can

be equivalently reformulated as $\min_{\pi\in\mathcal{A}} V^{'}(\pi)$ with $V^{'}(\pi) = \frac{\sum_{k=1}^{m} V_k^{'}(\pi)}{m}$ where $V_k^{'}(\pi) = \frac{\sum_{i=1}^{n}(s_i^{k,\pi})^2}{n}$, thus as a multi-matrix version of a multidimensional quadratic assignment problem (QAP).

2 Algorithms

In this section, we describe the two algorithms we propose for approximating the optimal rearrangement π^*.

2.1 Block Swapping Algorithm (BSA)

In the Block Swapping Algorithm (BSA), the rearranged matrices $\mathbf{X}^{k,\pi}$ are obtained from the matrices $\mathbf{X}^k$ by *swapping* in a selected block of columns (i.e., those for which $\gamma_j = 1$) the i_1-th element with the i_2-th element, leaving all other elements unaffected. Hence, we consider particular rearrangements $(\pi_1, \ldots, \pi_d)$ having the property that there exists $\gamma = (\gamma_1, \ldots, \gamma_d) \in \{0,1\}^d$ and $i_1, i_2 \in \{1, \ldots, n\}$ such that

i) for all $j \in \{1, \ldots, d\}$ such that $\gamma_j = 1$, it holds that $\pi_j(i_1) = i_2$, $\pi_j(i_2) = i_1$, and $\pi_j(i) = i$ for all $i \in \{1, \ldots, n\} \setminus \{i_1, i_2\}$;

ii) for all $j \in \{1, \ldots, d\}$ such that $\gamma_j = 0$, it holds that $\pi_j(i) = i$ for $i \in \{1, \ldots, n\}$.

We denote such a *block rearrangement* by $\pi(\gamma, i_1, i_2)$ with blocksize $\sum_{j=1}^{d} \gamma_j$. Application of the rearrangement $\pi(\gamma, i_1, i_2)$ on the matrices $\mathbf{X}^k$ changes $V(\pi)$ by the quantity

$$\Delta_{\pi(\gamma,i_1,i_2)} = \sum_{k=1}^{m} \left((s_{i_1}^k + \delta^k)^2 + (s_{i_2}^k - \delta^k)^2 - (s_{i_1}^k)^2 - (s_{i_2}^k)^2 \right), \tag{4}$$

where for $k = 1, \ldots, m$, we set

$$\delta^k = \sum_{j:\gamma_j=1} (x_{i_2 j}^k - x_{i_1 j}^k).$$

Hence, if

$$\Delta_{\pi(\gamma,i_1,i_2)} < 0, \tag{5}$$

then the application of the rearrangement $\pi(\gamma, i_1, i_2)$ to the matrices $\mathbf{X}^k$ leads to rearranged matrices $\mathbf{X}^{k,\pi}$ with a *strictly* smaller value for $V(\pi)$. The swapping condition (5) thus leads to the following algorithm.

Block Swapping Algorithm (BSA)

1. Let $\mathbf{X}^k$, $k = 1, \ldots, m$, be given $n \times d$ matrices and let n_{iter} be a positive integer.
2. Randomly select $\gamma \in \{0,1\}^d$ and $i_1, i_2 \in \{1, \ldots, n\}$; if the swapping condition (5) is fulfilled, then apply the rearrangement $\pi(\gamma, i_1, i_2)$ to the matrices $\mathbf{X}^k$ to obtain the matrices $\mathbf{X}^{k,\pi(\gamma,i_1,i_2)}$ and otherwise do nothing.
3. Let $\mathbf{X}^k = \mathbf{X}^{k,\pi(\gamma,i_1,i_2)}$ and repeat Step 2. n_{iter} times.
4. Output the final matrices $\mathbf{X}^k_*$ and the rearrangement π such that $\mathbf{X}^{k,\pi} = \mathbf{X}^k_*$.

At every iteration of BSA, the objective function is strictly reduced (when the swapping condition (5) is satisfied) or, otherwise, left unchanged. As a result, the BSA outputs a final rearrangement π with corresponding objective value $V(\pi)$ that can be expected to be close to the optimal synchronization value V^*. The BSA thus relies on the basic swapping idea that is inherent in the RA (Embrechts, Puccetti, and Rüschendorf (2013)) and adapts it to the case of multiple matrices. As there is ample evidence that approximations obtained by using the RA perform very well (see eg., Boudt, Jakobsons, and Vanduffel (2018)), we may thus expect that BSA is able to provide a good approximation of the optimal synchronization value V^*. This expectation is confirmed by the results of the numerical study we conduct in Section 3.

Remark 1. In this paper, we propose the BSA to obtain a rearrangement that minimizes the variance function $V(\pi)$. BSA is however a multi-purpose algorithm in that it can deal with the optimization of any Schur-convex function, i.e., formula (4) can be suitably adapted to the Schur-convex function at hand. Rearrangements obtained are in fact approximations for local minima w.r.t. Schur-convex order, i.e., a rearrangement that minimizes the variance function can be expected to yield also low values for other Schur-convex functions.

2.2 Customization of BSA (Cust BSA)

In a parallel development to this paper, Cornilly, Puccetti, Rüschendorf, and Vanduffel (2020) propose for the economic problem of fair allocation of indivisible goods the so-called Minimum Covariance Algorithm (MinCov) as an efficient method for dealing with the optimization problem (2). The MinCov algorithm, however, requires that the structure of the matrices $\mathbf{X}^k$ fulfils a specific condition, namely that there exist real numbers β^k_j, $k = 1, \ldots, m$, $j = 1, \ldots d$, such that

$$x^k_{ij} = \beta^k_j x^1_{ij}. \tag{6}$$

While condition (6) holds in the context of fair allocation, it does not hold for the synchronization problem with multiple instances we consider.

However, even when condition (6) does not hold, a slightly generalized version of MinCov can still be used as a pre-processor for BSA. We call this approach the Customized Block Swapping Algorithm (Cust BSA). To keep the paper self-contained, we briefly describe the MinCov algorithm and next we explain how it can be used as a pre-processor to BSA.

2.2.1 MinCov Algorithm

In this subsection, we specifically assume that the structure of the matrices $\mathbf{X}^k = (x_{ij}^k)$ fulfills the condition that there exist real numbers α_j^k and β_j^k, $k = 1, \ldots, m$, $j = 1, \ldots d$, such that

$$x_{ij}^k = \beta_j^k x_{ij}^1 + \alpha_j^k. \tag{7}$$

As compared to the original MinCov algorithm, we do not require that the coefficients α_j^k are equal to zero. Our observation is that the MinCov algorithm can also be used under the extended condition (7); this modest generalization is crucial for applying the linear regression step in the customization of BSA.

The MinCov algorithm uses rearrangements $\pi = (\pi_1, \ldots, \pi_d)$ having the property that there exist $j_1 \in \{1, \ldots, d\}$ such that $\pi_j(i) = i$ for $i \in \{1, \ldots, n\}$ and $j \neq j_1$. The matrices $\mathbf{X}^{k,\pi}$ are thus obtained from the matrices $\mathbf{X}^k$ by permuting (potentially all) elements in the j_1-th columns without affecting the others; we denote such rearrangements as $\pi(j_1)$. To explain the basic intuition of MinCov, for any π we interpret a $n \times d$ matrix $\mathbf{X}^{k,\pi}$ as a d-variate, discrete random vector where the rows represent possible realizations, each occurring with probability $1/n$.

Under this perspective, we can use for the formulation of problem (2) the probability language in which (co-)variance corresponds to the empirical (co-)variance, i.e., we consider the problem

$$\min_{\pi \in \mathcal{A}} \frac{\sum_{k=1}^m \operatorname{var}\left(S^{k,\pi}\right)}{m}, \tag{8}$$

in which

$$\operatorname{var}\left(S^{k,\pi}\right) = \frac{\sum_{i=1}^n (s_i^{k,\pi} - \overline{s}^k)^2}{n} \tag{9}$$

and where $S^{k,\pi} = (s_1^{k,\pi}, \ldots, s_n^{k,\pi})$. For $j_1 \in \{1, \ldots, d\}$, $S^{k,\pi}$ can be expressed as $S^{k,\pi} = X_{j_1}^{k,\pi} + S_{(-j_1)}^{k,\pi}$, where $X_{j_1}^{k,\pi}$ denotes the j_1-th column of $\mathbf{X}^{k,\pi}$ and

$$S_{(-j_1)}^{k,\pi} = \sum_{j \neq j_1} X_j^{k,\pi}.$$

It follows that an improvement in objective (8) is obtained if we determine a rearrangement $\pi(j_1)$

such that

$$\sum_{k=1}^{m} \operatorname{cov}\left(X_{j_1}^{k,\pi(j_1)}, S_{(-j_1)}^{k,\pi(j_1)}\right)$$

becomes as small as possible. As per assumption (7) we have that $X_{j_1}^{k,\pi(j_1)} = \beta_{j_1}^k X_{j_1}^{1,\pi(j_1)} + \alpha_{j_1}^k$; it follows that $\pi(j_1)$ should be chosen such that

$$\sum_{k=1}^{m} \operatorname{cov}\left(X_{j_1}^{k,\pi(j_1)}, S_{(-j_1)}^{k,\pi(j_1)}\right) = \sum_{k=1}^{m} \operatorname{cov}\left(X_{j_1}^{1,\pi(j_1)}, \beta_{j_1}^k S_{(-j_1)}^{k,\pi(j_1)}\right) = \operatorname{cov}\left(X_{j_1}^{1,\pi(j_1)}, \sum_{k=1}^{m} \beta_{j_1}^k S_{(-j_1)}^{k,\pi(j_1)}\right)$$

is minimum and thus such that $X_{j_1}^{1,\pi(j_1)}$ becomes oppositely ordered to $\sum_{k=1}^m \beta_{j_1}^k S_{(-j_1)}^{k,\pi(j_1)}$. These considerations lead to the following algorithm.

Minimum Covariance Algorithm (MinCov)

1. Let $\mathbf{X}^k$, $k = 1, \ldots, m$, be given $n \times d$ matrices and let n_{iter} be a positive integer.
2. Randomly select $j_1 \in \{1, \ldots, d\}$ and determine the rearrangement $\pi(j_1)$ such that $X_{j_1}^{1,\pi(j_1)}$ becomes oppositely ordered to $\sum_{k=1}^m \beta_{j_1}^k S_{(-j_1)}^{k,\pi(j_1)}$.
3. Re-label $\mathbf{X}^{k,\pi(j_1)}$ as $\mathbf{X}^k$ and repeat Step 2. n_{iter} times.
4. Output the final matrices $\mathbf{X}_*^k$ and the rearrangement π such that $\mathbf{X}^{k,\pi} = \mathbf{X}_*^k$.

2.2.2 Customization

The idea of the customization of BSA is to first transform the initial matrices $\mathbf{X}^k$ by approximating them by a linear regression which yields transformed matrices $\mathbf{Y}^k$ that satisfy condition (7), and to apply next MinCov to the $\mathbf{Y}^k$. The solution, i.e., a rearrangement, is then applied to the original matrices $\mathbf{X}^k$ after which BSA is applied. We formulate the following algorithm.

Customized BSA (Cust BSA)

1. Let $\mathbf{X}^k$, $k = 1, \ldots, m$, be given $n \times d$ matrices and let n_{iter} be a positive integer.
2. Randomly select $j_1 \in \{1, \ldots, d\}$ and, for each $k = 2, \ldots, m$, perform a linear regression on the n data pairs $(x^1_{ij_1}, x^k_{ij_1})$ to obtain $y^k_{ij_1} = \beta^k_{j_1} x^1_{ij_1} + \alpha^k_{j_1}$ for appropriate regression coefficients $\beta^k_{j_1}$ and $\alpha^k_{j_1}$.
3. Define the matrices $\mathbf{Y}^k = (y^k_{ij})$ in which $y^k_{ij} = x^k_{ij}$, $j \neq j_1$, and $y^k_{ij_1} = \beta^k_{j_1} x^1_{ij_1} + \alpha^k_{j_1}$.
4. Apply a MinCov iteration to the matrices $\mathbf{Y}^k$ (i.e., apply Step 2. of MinCov to the j_1-th column of $\mathbf{Y}^k$) and apply the obtained rearrangement to the matrices $\mathbf{X}^k$.
5. Apply BSA to the output matrices obtained in Step 4. using n_{iter} iterations.
6. Output the final matrices $\mathbf{X}^k_*$ and the rearrangement π such that $\mathbf{X}^{k,\pi} = \mathbf{X}^k_*$.

Remark 2.1 (Repetition of pre-processing)**.** Steps 2-4 perform the pre-processing of Cust BSA and can also be applied several times, each time selecting a different column number j_1.

Remark 2.2 (Generalization of the pre-processing step using blocks)**.** The use of MinCov as a pre-processor is also possible via blocks. Randomly select $\delta \in \{0,1\}^d$ and replace all columns for which $\delta_j = 1$ by a single column with values $y^k_{i1} = \sum_{j=1}^m \delta_j x^k_{ij}$. On these matrices, apply the MinCov pre-processing Steps 2-4 based on the first column, resulting in a permutation π only for the column containing the block row sums. This permutation is then applied to each of the columns of the original matrices $\mathbf{X}^k$ selected by $\delta_j = 1$. As stated above, this procedure can be repeated, each time selecting a different random block. After this pre-processing, one applies BSA.

3 Numerical study of BSA and Cust BSA

We investigate the quality of BSA and Cust BSA for the synchronization problem (generalized ALCS problem) where labour costs (production times) are described by various problem instances $\mathbf{X}^k$. Recall that a minimum variance solution $\pi^* \in \mathcal{A}$ yields an optimal synchronization under which the production costs (time lenghts) of the different production lines are as equal as possible across *all* scenarios. To this end, construct initial matrices $\mathbf{X}^k$ by independently drawing the (x^k_{ij}), $i = 1, \ldots, n$, from the joint distribution $\mathbf{F} \sim (Z^k_j)$, where (Z^k_j) is a multivariate normally distributed random vector with standard normal marginal distributions $F^k_j \sim N(0,1)$ and correlations $\rho(Z^{k_1}_{j_1}, Z^{k_2}_{j_2}) = \rho\delta_{j_1,j_2}$if $k_1 \neq k_2$ and $\rho(Z^{k_1}_{j_1}, Z^{k_2}_{j_2}) = \delta_{j_1,j_2}$, if $k_1 = k_2$. Here, $\delta_{\cdot,\cdot}$ denotes the Kronecker delta. Roughly speaking, labour times of different workers are uncorrelated, whereas for a given worker the labour times across the various possible scenarios are correlated.

In the applications to follow, we let the number of rows be $n = 10, 100, 1000$, the number of columns $d = 5, 50, 500$, the number of matrices $m = 5, 10, 20$, and $\rho = 0.1, 0.5, 0.8$. For each setting, we generate 1000 random initial configurations (that is, 1000 different sets of m matrices $\mathbf{X}^k$) and apply each time the BSA and Cust BSA algorithms. As our extended synchronization problem has not yet been considered in the literature, we use as a reference method a simple *randomized* approach, called RandSelect, which consists in applying a randomly selected rearrangement π iteratively only if the objective $V(\pi)$ decreases. In all our numerical experiments we fix the maximum number of iterations $n_{iter} = 10^6$ for each algorithm, but we stop running a given algorithm when no improvement is observed for a given number of consecutive iterations, which is set equal to $\max(2d, 2n)$. In the latter case we say that the algorithm has converged. Note that in all numerical experiments we conducted, convergence was always obtained.

For Cust BSA, in the pre-processing phase (Steps 2-4.) we run MinCov three times with block version using an expected block size of $d/2$, i.e., each column is randomly selected with probability $1/2$ for the first block, as described in Remark 2.2. Hence, after the pre-processing we have $n_{iter} - 3$ iterations that are left for running BSA. In the second phase (Step 5.), the algorithm proceeds with BSA using expected blocksizes $d/2, d/4$, and $d/8$ (probabilities for a column to be selected are $1/2$, $1/4$, and $1/8$, respectively) and running $n \log(5 + d)$ (rounded to the nearest integer) iterations for each blocksize, before continuing using blocksizes that are exactly equal to 1 and running iterations until convergence or until the total budget of n_{iter} iterations is consumed. All computations were run on a 2014 MacBook Air with 1.7 GHz Intel Core i7 processor.

3.1 Accuracy of the algorithms

In Table 1, we consider $\rho = 0.5$ and provide for all algorithms summary statistics related to the objective function $V(\pi)$. We also provide these summary statistics for the initial (not rearranged) matrices, so that we can assess the extent to which the variability among row sums decreases when applying the various algorithms. The results clearly show that BSA performs reasonably well, in particular when the value for d is rather small, but Cust BSA is clearly outperforming, and moreover is able to find configurations with low objective value V^*. Specifically, for all values of n, d and n, Cust BSA is always able to significantly improve an initial (randomly chosen) allocation. Moreover, the degree of improvement (over an initial allocation) of Cust BSA is clearly increasing in d, and, for a given value of m, V^* tends to be the lowest for high values of d or n. This last observation is partially driven by the fact that when increasing d or n, it is more likely that there are rearrangements that yield stable row sums and Cust BSA is able to find these. We also find, as expected, a poor performance of RandSelect showing that a purely randomized approach is not the appropriate way to go forward. Moreover, as RandSelect requires that in each iteration all elements in all matrices are assessed, it quickly suffers from the curse of dimensionality. NA's in the tables to follow indicate that running RandSelect is computationally too expensive.

		$d=5$			$d=50$			$d=500$		
		min.	med.	max.	min.	med.	max.	min.	med.	max.
Panel A: $m=5$										
$n=10$	Initial	3.24	5.43	8.49	32.71	53.82	83.47	318.22	533.11	830.86
	RandSelect	1.63	2.92	4.80	12.59	22.78	37.69	NA	NA	NA
	BSA	0.67	1.22	2.03	2.63	4.68	7.69	9.18	16.37	26.75
	Cust BSA	0.63	1.15	1.93	0.33	0.60	1.00	0.09	0.16	0.26
$n=100$	Initial	4.34	5.05	5.80	43.19	50.21	57.86	436.43	504.69	580.02
	RandSelect	3.37	3.94	4.57	33.13	38.70	44.65	NA	NA	NA
	BSA	0.41	0.48	0.56	3.77	4.42	5.17	18.03	21.28	24.79
	Cust BSA	0.36	0.43	0.50	0.44	0.52	0.61	0.17	0.21	0.24
$n=1000$	Initial	4.78	5.00	5.22	47.77	50.03	52.31	478.61	500.59	523.13
	RandSelect	NA	NA	NA	NA	NA	NA	NA	NA	NA
	BSA	0.19	0.20	0.21	1.79	1.89	1.98	18.03	18.97	19.94
	Cust BSA	0.16	0.17	0.18	0.17	0.18	0.19	0.17	0.18	0.19
Panel B: $m=10$										
$n=10$	Initial	2.64	5.38	9.55	26.52	53.75	95.53	257.93	525.18	933.93
	RandSelect	1.41	3.14	5.92	11.73	25.97	50.07	NA	NA	NA
	BSA	0.72	1.65	3.19	3.91	8.83	16.95	21.29	46.84	88.26
	Cust BSA	0.70	1.57	3.07	0.69	1.59	3.11	0.25	0.57	1.07
$n=100$	Initial	4.14	5.03	6.04	41.00	50.14	60.40	414.86	504.05	606.88
	RandSelect	3.28	4.05	4.91	32.30	39.78	48.27	NA	NA	NA
	BSA	0.72	0.90	1.10	6.70	8.37	10.27	42.56	52.87	64.99
	Cust BSA	0.66	0.83	1.01	1.10	1.37	1.69	0.53	0.67	0.82
$n=1000$	Initial	4.71	5.00	5.30	47.11	50.02	53.10	470.86	499.98	530.10
	RandSelect	NA	NA	NA	NA	NA	NA	NA	NA	NA
	BSA	0.49	0.53	0.56	4.64	4.97	5.31	46.31	49.55	52.90
	Cust BSA	0.44	0.47	0.50	0.54	0.58	0.62	0.56	0.59	0.64
Panel C: $m=20$										
$n=10$	Initial	2.20	5.35	10.48	21.98	53.82	106.04	219.19	535.86	1042.86
	RandSelect	NA	NA	NA	NA	NA	NA	NA	NA	NA
	BSA	0.72	1.99	4.39	4.66	12.94	28.22	31.30	85.53	185.74
	Cust BSA	0.70	1.95	4.32	1.22	3.50	7.70	0.53	1.47	3.18
$n=100$	Initial	3.98	5.04	6.27	39.46	50.19	62.65	395.61	501.66	624.30
	RandSelect	NA	NA	NA	NA	NA	NA	NA	NA	NA
	BSA	1.00	1.31	1.68	9.36	12.29	15.72	69.35	90.95	116.96
	Cust BSA	0.94	1.23	1.59	2.22	2.93	3.78	1.28	1.69	2.17
$n=1000$	Initial	4.65	5.00	5.37	46.49	50.04	53.77	464.37	500.09	536.51
	RandSelect	NA	NA	NA	NA	NA	NA	NA	NA	NA
	BSA	0.84	0.92	1.00	8.00	8.69	9.43	79.84	86.60	93.87
	Cust BSA	0.78	0.85	0.92	1.28	1.40	1.51	1.45	1.58	1.72

Table 1: We compute the m **variances** of the row sums of m initial matrices as well as after application of various algorithms. We store the minimum, median, maximum, and average variance ($V(\pi)$). Each statistic shown is the average of the individual summary statistics over 1000 random initial configurations. The correlation parameter is fixed to $\rho = 0.5$.

To the best of our knowledge, there are no other benchmark algorithms available in the literature for the generalized ALCS problem and no explicit solutions nor good bounds are readily available. A notable exception, however, occurs when there is only one matrix to rearrange (situation of one problem instance, i.e., when $m = 1$). In this case, it can be shown that for the set-up we consider in the numerical study there exists, at least in the case $n \to \infty$, a rearrangement π such that $V(\pi) = 0$. Therefore, it is of interest to assess the extent by which the algorithms we propose are able to find this optimal solution. Table 2 shows that in particular Cust BSA is able to closely approximate the optimal rearrangement, indicating that the two algorithms can be expected to also approximate optimal solutions when there are several problem instances ($m > 1$).

		$d = 5$			$d = 50$			$d = 500$		
		min.	med.	max.	min.	med.	max.	min.	med.	max.
$n = 10$	Initial	5.57	5.57	5.57	55.40	55.40	55.40	565.41	565.41	565.41
	RandSelect	1.71	1.71	1.71	10.19	10.19	10.19	56.54	56.54	56.54
	BSA	0.19	0.19	0.19	0.07	0.07	0.07	0.01	0.01	0.01
	Cust BSA	0.09	0.09	0.09	0.01	0.01	0.01	0.00	0.00	0.00
$n = 100$	Initial	5.07	5.07	5.07	51.01	51.01	51.01	502.41	502.41	502.41
	RandSelect	3.36	3.36	3.36	32.20	32.20	32.20	297.77	297.77	297.77
	BSA	0.01	0.01	0.01	0.05	0.05	0.05	0.02	0.02	0.02
	Cust BSA	0.00	0.00	0.00	0.01	0.01	0.01	0.00	0.00	0.00
$n = 1000$	Initial	5.01	5.01	5.01	50.05	50.05	50.05	501.28	501.28	501.28
	RandSelect	4.31	4.31	4.31	42.52	42.52	42.52	424.45	424.45	424.45
	BSA	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.01	0.01
	Cust BSA	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Table 2: The same statistics (w.r.t. the variance) as in Table 1 for $m = 1$.

3.2 Stability

In order to gain further insight in the ability of the algorithms to improve the initial configuration, we show in Table 3 and 4 summary statistics for the *range* of the row sums (production times) that exist among the different assembly lines after application of the different algorithms. Clearly, Cust BSA leads to very stable results along this dimension across the various matrices, proving itself to be a useful tool when dealing with the generalized ALCS problem with uncertain labour costs. For instance, when $(n, d, m) = (1000, 500, 10)$ we obtain that Cust BSA is able to reduce by a factor of 30 the average range of a matrix resulting from a random configuration (i.e., from 144.34 to 4.83). Also BSA leads to satisfactory results. These observations confirm the statements in Remark 2.1 that rearrangements with (approximate) minimum variance also show low values with respect to other Schur-convex functions like for instance the range. Note that BSA could be adapted to minimizing the range instead of the variance function in which case the results would be further improved.

		$d=5$			$d=50$			$d=500$		
		min.	med.	max.	min.	med.	max.	min.	med.	max.
Panel A: $m=5$										
$n=10$	Initial	5.15	6.86	8.84	16.33	21.65	27.77	50.68	67.95	87.84
	RandSelect	3.70	5.05	6.69	10.21	14.16	18.63	NA	NA	NA
	BSA	2.34	3.25	4.33	4.66	6.36	8.36	8.71	11.89	15.47
	Cust BSA	2.27	3.15	4.20	1.64	2.28	3.02	0.86	1.18	1.53
$n=100$	Initial	9.80	11.16	12.78	31.00	35.21	40.23	98.21	111.62	127.62
	RandSelect	8.66	9.88	11.35	27.16	30.86	35.46	NA	NA	NA
	BSA	2.99	3.41	3.93	9.01	10.21	11.69	19.61	22.26	25.35
	Cust BSA	2.81	3.21	3.71	3.10	3.55	4.10	1.94	2.21	2.52
$n=1000$	Initial	13.31	14.45	15.82	42.03	45.55	50.01	133.22	144.44	158.68
	RandSelect	NA	NA	NA	NA	NA	NA	NA	NA	NA
	BSA	2.61	2.82	3.09	7.84	8.43	9.18	24.85	26.72	29.02
	Cust BSA	2.44	2.64	2.92	2.44	2.63	2.87	2.43	2.63	2.87
Panel B: $m=10$										
$n=10$	Initial	4.62	6.85	9.48	14.58	21.66	29.94	45.63	67.37	94.11
	RandSelect	3.42	5.27	7.49	9.86	15.12	21.81	NA	NA	NA
	BSA	2.43	3.80	5.48	5.69	8.80	12.54	13.26	20.24	28.60
	Cust BSA	2.40	3.72	5.37	2.38	3.73	5.38	1.43	2.23	3.16
$n=100$	Initial	9.43	11.14	13.31	29.70	35.16	42.08	94.35	111.49	133.26
	RandSelect	8.45	10.02	12.02	26.38	31.30	37.60	NA	NA	NA
	BSA	3.90	4.64	5.55	11.95	14.12	16.80	30.00	35.39	42.09
	Cust BSA	3.77	4.46	5.36	4.85	5.77	6.97	3.38	4.00	4.81
$n=1000$	Initial	13.02	14.42	16.32	41.13	45.62	51.67	130.16	144.34	162.95
	RandSelect	NA	NA	NA	NA	NA	NA	NA	NA	NA
	BSA	4.11	4.55	5.12	12.51	13.78	15.43	39.50	43.45	48.72
	Cust BSA	3.91	4.32	4.88	4.33	4.78	5.40	4.37	4.83	5.45
Panel C: $m=20$										
$n=10$	Initial	4.20	6.83	10.03	13.22	21.71	32.03	41.74	68.45	100.48
	RandSelect	NA	NA	NA	NA	NA	NA	NA	NA	NA
	BSA	2.41	4.20	6.49	6.17	10.68	16.33	16.02	27.43	41.91
	Cust BSA	2.39	4.15	6.43	3.17	5.55	8.59	2.07	3.59	5.52
$n=100$	Initial	9.14	11.15	13.89	28.72	35.18	43.72	91.00	111.23	138.42
	RandSelect	NA	NA	NA	NA	NA	NA	NA	NA	NA
	BSA	4.57	5.63	7.04	13.94	17.21	21.41	37.95	46.74	58.08
	Cust BSA	4.45	5.47	6.85	6.84	8.45	10.64	5.18	6.40	8.02
$n=1000$	Initial	12.73	14.40	16.75	40.26	45.55	53.08	127.42	144.11	167.84
	RandSelect	NA	NA	NA	NA	NA	NA	NA	NA	NA
	BSA	5.37	6.05	7.01	16.44	18.47	21.34	51.80	58.31	67.58
	Cust BSA	5.16	5.83	6.75	6.64	7.50	8.71	7.08	8.00	9.32

Table 3: We compute the m **ranges** of the row sums of m initial matrices as well as after application of various algorithms. We store the minimum, median, maximum, and average range. Each statistic shown is the average of the individual summary statistics over 1000 random initial configurations. The correlation parameter is fixed to $\rho = 0.5$.

3.3 Time complexity

We investigate in this subsection the run times and convergence properties of the algorithms as well their behaviour across the iterations. Table 5 shows that BSA and in particular Cust BSA are fast algorithms that are also able to efficiently deal with high-dimensional set-ups. For example, when rearranging 20 matrices of 1000 rows and 500 columns each (i.e., when $(n, d, m) = (1000, 500, 20)$), Cust BSA reaches convergence after 15 seconds.

Table 6 shows the number of iterations needed to obtain convergence. As compared to BSA, Cust BSA uses more iterations to reach convergence and the pre-processing step of Cust BSA is

		$d=5$			$d=50$			$d=500$		
		min.	med.	max.	min.	med.	max.	min.	med.	max.
$n=10$	Initial	6.87	6.87	6.87	21.78	21.78	21.78	69.54	69.54	69.54
	RandSelect	3.85	3.85	3.85	9.44	9.44	9.44	22.28	22.28	22.28
	BSA	1.20	1.20	1.20	0.73	0.73	0.73	0.22	0.22	0.22
	Cust BSA	0.86	0.86	0.86	0.23	0.23	0.23	0.02	0.02	0.02
$n=100$	Initial	11.27	11.27	11.27	35.59	35.59	35.59	111.74	111.74	111.74
	RandSelect	9.15	9.15	9.15	28.39	28.39	28.39	86.20	86.20	86.20
	BSA	0.40	0.40	0.40	1.18	1.18	1.18	0.76	0.76	0.76
	Cust BSA	0.39	0.39	0.39	0.59	0.59	0.59	0.07	0.07	0.07
$n=1000$	Initial	14.50	14.50	14.50	45.91	45.91	45.91	144.88	144.88	144.88
	RandSelect	13.44	13.44	13.44	42.26	42.26	42.26	133.66	133.66	133.66
	BSA	0.09	0.09	0.09	0.25	0.25	0.25	0.79	0.79	0.79
	Cust BSA	0.09	0.09	0.09	0.09	0.09	0.09	0.24	0.24	0.24

Table 4: The same statistics (w.r.t. the range) as in Table 3 for the case $m = 1$.

	$d=5$			$d=50$			$d=500$		
	$n=10$	$n=100$	$n=1000$	$n=10$	$n=100$	$n=1000$	$n=10$	$n=100$	$n=1000$
Panel A: $m=5$									
RandSelect	0.0057	0.0255	NA	0.0136	0.1063	NA	NA	NA	NA
BSA	0.0123	0.0113	0.2299	0.0109	0.0410	1.7123	0.2278	1.1572	33.5891
Cust BSA	0.0187	0.0133	0.1773	0.0130	0.0428	0.7838	0.0761	0.4110	6.1285
Panel B: $m=10$									
RandSelect	0.0052	0.0333	NA	0.0167	0.1828	NA	NA	NA	NA
BSA	0.0104	0.0117	0.2714	0.0113	0.0473	2.9160	0.2614	2.1298	48.4079
Cust BSA	0.0152	0.0178	0.2006	0.0125	0.0506	1.1880	0.0943	0.6216	9.3419
Panel C: $m=20$									
RandSelect	NA	NA	NA	NA	NA	NA	NA	NA	NA
BSA	0.0102	0.0172	0.4153	0.0141	0.0656	7.1237	0.3596	4.9473	78.7507
Cust BSA	0.0217	0.0276	0.3113	0.0256	0.0635	2.2472	0.1331	1.0985	15.0317

Table 5: Computation times to reach convergence for a combination of n, m, d parameters (average over 1000 random initial configurations, time in seconds).

somewhat time costly. Nevertheless, each of the following iterations typically require less time than in the case of BSA leading to a faster convergence overall. The reason why Cust BSA is typically faster per iteration is due to the fact that in each iteration BSA deals with $d/2$ columns on average, whereas for Cust BSA the number of columns that require manipulation decreases in a structured way along the iterations.

	$d=5$			$d=50$			$d=500$		
	$n=10$	$n=100$	$n=1000$	$n=10$	$n=100$	$n=1000$	$n=10$	$n=100$	$n=1000$
Panel A: $m=5$									
RandSelect	34	360	NA	179	354	NA	NA	NA	NA
BSA	106	5501	328360	520	5402	316438	4934	24819	310564
Cust BSA	98	5653	336200	912	10332	420764	6327	39902	476747
Panel B: $m=10$									
RandSelect	36	357	NA	178	359	NA	NA	NA	NA
BSA	106	5589	329646	518	5382	310170	4826	24960	308872
Cust BSA	102	5765	340791	1186	13915	567468	8236	57327	686789
Panel C: $m=20$									
RandSelect	NA	NA	NA	NA	NA	NA	NA	NA	NA
BSA	108	5605	344864	523	5575	324867	4926	25466	322290
Cust BSA	101	5993	362787	1496	18653	822628	12255	89764	987959

Table 6: Average number of iterations to reach convergence for a combination of n, m, d parameters (average over 1000 random initial configurations). The maximum of $n_{iter} = 10^6$ iterations has never been reached.

We also show in Figure 1 for some selected combinations of (n, d, m) the extent by which the objective $V(\pi)$ decreases as a function of the number of iterations performed. We observe that for all algorithms the objective value first quickly decreases with a much slower decline in later iterations. In particular, Cust BSA appears to achieve the fastest reduction of the objective value indicating that the pre-processing step is very useful. This fast initial reduction is also clear from Table 7, which shows for all combinations for (n, d, m) the improvement in the objective value for Cust BSA after the preprocessing step. Figure 1 shows that also in later iterations Cust BSA performs slightly better than BSA suggesting that selecting the block sizes in a structured manner has positive impact on performance.

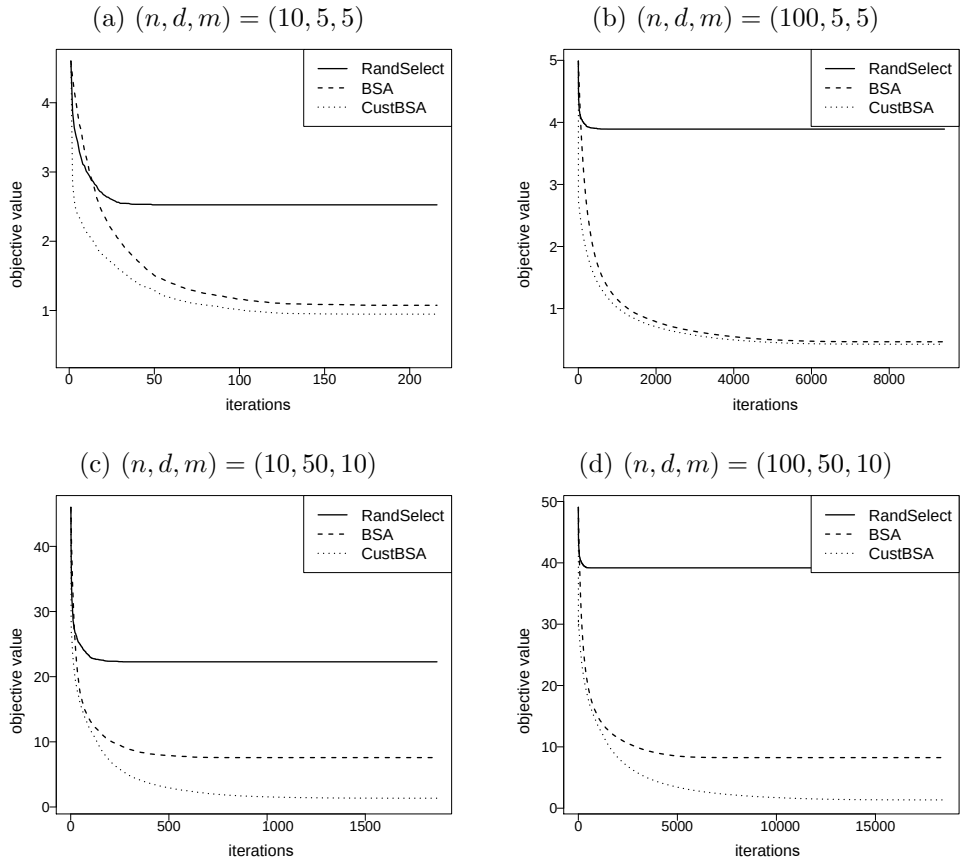

Figure 1: We show for different values of (n, d, m) how the objective value $V(\pi)$ evolves when increasing the number of iterations (average over 1000 random initial configurations).

	$d = 5$			$d = 50$			$d = 500$		
	$n = 10$	$n = 100$	$n = 1000$	$n = 10$	$n = 100$	$n = 1000$	$n = 10$	$n = 100$	$n = 1000$
Panel A: $m = 5$									
$m = 5$	53.31	46.30	45.25	53.91	45.15	44.39	53.51	45.30	44.41
$m = 10$	49.03	40.17	39.19	49.01	38.95	38.33	47.69	39.22	38.17
$m = 20$	46.56	37.31	36.43	46.76	36.19	35.49	45.54	36.27	35.33

Table 7: Improvement in objective value expressed as a percentage after the pre-processing steps of Cust BSA for a combination of n, m, d parameters (average over 1000 random initial configurations).

3.4 Robustness with respect to the correlation parameter ρ

In Tables 8–11 we assess the performance of the algorithms for other values for the correlation parameter ρ. First, in Table 8 and Table 9 we consider $\rho = 0.8$ and observe that the algorithms BSA and in particular Cust BSA deliver high quality solutions, even outperforming those presented in Table 1 and Table 3, respectively (in which $\rho = 0.5$). These figures are to be expected as higher values for ρ – reflecting the strength of dependence among labour costs (production times) of a worker – imply that the m scenarios are more similar. In the limiting case, when $\rho = 1$, the m matrices coincide and the setting reduces to the case $m = 1$. Second, in Table 10 and Table 11 we assess the case in which $\rho = -0.1$. We observe that also in this case Cust BSA continues to deliver excellent results.

3.5 Robustness with respect to the degree of variability of production times

Finally, we also consider the case in which the production costs for a certain task present more variability. In the basic set-up, the (x_{ij}^k), $i = 1, \ldots, n$, were drawn from the joint distribution $\mathbf{F} \sim (Z_j^k)$, where (Z_j^k) was a multivariate random vector with standard normal marginal distributions $F_j^k \sim N(0, 1)$ and a Gaussian copula with a single correlation parameter $\rho = 0.5$. Here, we preserve the Gaussian dependence but assume the margins $F_j^k \sim t_5$ where t_5 stands for a t-distribution with five degrees of freedom, thus leading to more variability than in the case of a standard normal distribution for the margins. The results in Table 12 and Table 13 show that also in this case Cust BSA is outperforming and is still able to find configurations with low variance.

4 Conclusions

In this paper, we study a class of synchronization problems that can be seen as generalizations of the classical ALCS problem (to the case of multiple scenarios) as well as of several further data association and tracking problems that appeared in the literature on multidimensional assignment problems. To the best of our knowledge this class of problems has not yet been considered and no algorithms are available (unless there is only one scenario). We introduce the BSA algorithm for solving this kind of synchronization problems and analyze its properties. It turns out that BSA does not suffer from the curse of dimensionality in that it can deal with high dimensions, i.e., when there are a large numbers of jobs, workers and problem instances (scenarios).

We propose a suitable pre-processing step, which builds on a linear regression combined with a recently introduced algorithm called MinCov. The resulting algorithm Cust BSA provides a further improvement of BSA. In general, the algorithms yield – similar to the Rearrangement Algorithm (RA) in the single instance case – fast solutions that are of good quality. Cust BSA is designed for dealing with the variance function as objective (see equation (3)) and significantly

		$d=5$			$d=50$			$d=500$		
		min.	med.	max.	min.	med.	max.	min.	med.	max.
Panel A: $m=5$										
$n=10$	Initial	4.09	5.74	7.67	39.64	55.85	75.85	399.56	570.96	775.91
	RandSelect	1.34	2.28	3.46	9.48	15.93	25.39	NA	NA	NA
	BSA	0.38	0.72	1.14	1.38	2.66	4.30	4.96	8.86	14.51
	Cust BSA	0.34	0.68	1.17	0.22	0.39	0.68	0.05	0.08	0.14
$n=100$	Initial	4.59	5.06	5.59	45.48	50.28	55.58	467.37	515.77	565.84
	RandSelect	3.21	3.58	3.99	31.04	34.47	39.14	NA	NA	NA
	BSA	0.22	0.26	0.31	2.04	2.39	2.77	9.26	10.88	12.90
	Cust BSA	0.20	0.24	0.28	0.27	0.31	0.36	0.10	0.11	0.13
$n=1000$	Initial	4.83	4.99	5.13	48.25	49.95	51.51	484.06	499.20	514.73
	RandSelect	NA	NA	NA	NA	NA	NA	NA	NA	NA
	BSA	0.10	0.11	0.11	0.94	0.99	1.05	9.38	9.86	10.37
	Cust BSA	0.09	0.09	0.10	0.09	0.10	0.10	0.09	0.10	0.10
Panel B: $m=10$										
$n=10$	Initial	3.52	5.77	8.41	34.97	55.33	83.50	333.99	534.14	781.24
	RandSelect	1.15	2.37	4.13	8.45	17.80	32.46	NA	NA	NA
	BSA	0.38	0.86	1.68	1.85	4.30	8.51	9.95	22.49	42.92
	Cust BSA	0.38	0.82	1.59	0.40	0.97	1.91	0.14	0.28	0.54
$n=100$	Initial	4.51	5.12	5.85	44.10	50.41	57.79	444.20	504.96	579.47
	RandSelect	3.11	3.66	4.24	30.45	35.72	41.29	NA	NA	NA
	BSA	0.35	0.44	0.54	3.27	4.10	5.00	20.38	25.53	31.23
	Cust BSA	0.32	0.40	0.50	0.64	0.80	0.98	0.28	0.34	0.43
$n=1000$	Initial	4.78	4.99	5.19	48.16	50.24	52.31	478.45	500.22	521.46
	RandSelect	NA	NA	NA	NA	NA	NA	NA	NA	NA
	BSA	0.23	0.25	0.27	2.20	2.36	2.53	21.94	23.38	24.93
	Cust BSA	0.21	0.22	0.24	0.28	0.30	0.32	0.28	0.30	0.33
Panel C: $m=20$										
$n=10$	Initial	3.05	5.44	8.71	31.00	55.42	88.38	309.18	549.91	880.59
	RandSelect	NA	NA	NA	NA	NA	NA	NA	NA	NA
	BSA	0.38	1.01	2.22	2.14	5.86	12.93	13.13	37.47	82.56
	Cust BSA	0.34	0.96	2.10	0.76	2.02	4.53	0.26	0.75	1.60
$n=100$	Initial	4.41	5.20	6.02	43.12	50.34	58.82	420.95	495.58	577.68
	RandSelect	NA	NA	NA	NA	NA	NA	NA	NA	NA
	BSA	0.45	0.60	0.78	4.39	5.72	7.28	31.07	40.35	52.27
	Cust BSA	0.44	0.58	0.73	1.23	1.60	2.06	0.64	0.85	1.10
$n=1000$	Initial	4.73	4.99	5.24	47.91	50.33	52.87	473.58	497.84	521.59
	RandSelect	NA	NA	NA	NA	NA	NA	NA	NA	NA
	BSA	0.38	0.41	0.45	3.58	3.91	4.23	35.50	38.61	41.87
	Cust BSA	0.35	0.38	0.42	0.63	0.69	0.75	0.83	0.90	0.98

Table 8: The same statistics (w.r.t. the variance) as in Table 1 for correlation parameter $\rho = 0.8$.

outperforms BSA. By contrast BSA can be adapted to any given Schur-convex function and is thus more generally applicable.

As a special case of the BSA, one might only consider blocks of a single column. Under this modification, the BSA reduces in the case of two matrices (i.e., when $m = 2$) to the swapping Algorithm (SA), introduced in Rüschendorf and Rachev (1990) as an algorithm for the computation of the Wasserstein distance between two probability distributions, and investigated in detail in Puccetti (2017). It is our experience by numerical experiments that the BSA converges faster than SA, which is of particular importance in high-dimensional set-ups.

		$d=5$			$d=50$			$d=500$		
		min.	med.	max.	min.	med.	max.	min.	med.	max.
Panel A: $m=5$										
$n=10$	Initial	5.76	7.03	8.42	17.88	22.01	26.21	56.10	70.78	84.34
	RandSelect	3.37	4.47	5.76	8.89	11.91	15.56	NA	NA	NA
	BSA	1.76	2.46	3.23	3.38	4.79	6.22	6.33	8.72	11.44
	Cust BSA	1.68	2.44	3.29	1.33	1.83	2.47	0.63	0.86	1.13
$n=100$	Initial	10.00	11.22	12.45	31.66	34.86	38.65	101.15	113.35	125.23
	RandSelect	8.53	9.40	10.63	26.28	29.14	33.34	NA	NA	NA
	swapping	2.08	2.41	2.78	2.46	2.75	3.17	1.41	1.62	1.87
	BSA	2.20	2.50	2.88	6.65	7.46	8.55	14.15	15.89	18.16
	Cust BSA	2.10	2.41	2.80	2.42	2.75	3.21	1.46	1.65	1.90
$n=1000$	Initial	13.45	14.48	15.53	42.51	45.59	48.95	135.57	145.99	157.84
	RandSelect	NA	NA	NA	NA	NA	NA	NA	NA	NA
	BSA	1.91	2.06	2.26	5.70	6.15	6.67	17.77	19.16	21.13
	Cust BSA	1.77	1.92	2.11	1.79	1.95	2.10	1.79	1.95	2.13
Panel B: $m=10$										
$n=10$	Initial	5.28	7.07	8.84	16.66	21.85	27.92	51.18	67.46	85.76
	RandSelect	3.04	4.55	6.20	8.39	12.54	17.75	NA	NA	NA
	BSA	1.73	2.73	3.95	3.86	6.12	8.94	9.00	14.06	19.93
	Cust BSA	1.74	2.71	3.92	1.80	2.93	4.23	1.06	1.59	2.21
$n=100$	Initial	10.01	11.43	13.08	30.84	35.17	40.32	96.81	111.34	128.04
	RandSelect	8.17	9.53	11.24	25.36	29.49	34.35	NA	NA	NA
	BSA	2.76	3.25	3.88	8.36	9.91	11.87	20.75	24.56	29.27
	Cust BSA	2.61	3.14	3.73	3.74	4.42	5.35	2.41	2.90	3.50
$n=1000$	Initial	13.25	14.50	16.16	41.96	46.01	51.25	131.09	144.51	160.88
	RandSelect	NA	NA	NA	NA	NA	NA	NA	NA	NA
	BSA	2.85	3.14	3.51	8.65	9.51	10.66	27.11	29.91	33.75
	Cust BSA	2.70	2.99	3.38	3.12	3.45	3.90	3.14	3.47	3.94
Panel C: $m=20$										
$n=10$	Initial	4.90	6.85	9.17	15.37	21.78	29.14	48.77	68.81	91.86
	RandSelect	NA	NA	NA	NA	NA	NA	NA	NA	NA
	BSA	1.75	2.98	4.70	4.20	7.19	11.09	10.48	18.20	27.98
	Cust BSA	1.66	2.90	4.49	2.47	4.19	6.57	1.44	2.56	3.95
$n=100$	Initial	9.73	11.50	13.63	30.24	35.43	41.75	94.15	111.87	132.64
	RandSelect	NA	NA	NA	NA	NA	NA	NA	NA	NA
	BSA	3.10	3.81	4.76	9.53	11.75	14.54	25.56	31.28	39.01
	Cust BSA	3.05	3.77	4.68	5.04	6.25	7.87	3.70	4.57	5.73
$n=1000$	Initial	13.05	14.49	16.32	41.07	45.81	51.86	129.44	143.48	162.96
	RandSelect	NA	NA	NA	NA	NA	NA	NA	NA	NA
	BSA	3.60	4.06	4.71	10.99	12.42	14.29	34.76	38.87	45.16
	Cust BSA	3.49	3.91	4.53	4.67	5.29	6.12	5.34	6.08	7.02

Table 9: The same statistics (w.r.t. the variance) as in Table 3 for correlation parameter $\rho = 0.8$.

		$d=5$			$d=50$			$d=500$		
		min.	med.	max.	min.	med.	max.	min.	med.	max.
Panel A: $m=5$										
$n=10$	Initial	2.84	5.18	8.90	29.84	52.61	87.47	289.16	538.70	890.72
	RandSelect	1.87	3.38	5.56	15.10	28.02	46.37	NA	NA	NA
	BSA	0.92	1.61	2.64	3.24	5.84	9.51	12.78	22.42	37.39
	Cust BSA	0.83	1.47	2.36	0.44	0.77	1.26	0.11	0.20	0.32
$n=100$	Initial	4.31	5.03	5.91	42.20	49.84	58.54	427.72	505.85	592.30
	RandSelect	3.58	4.23	4.91	35.53	41.67	48.71	NA	NA	NA
	BSA	0.54	0.64	0.74	5.05	5.91	6.85	23.57	28.07	32.53
	Cust BSA	0.48	0.57	0.66	0.56	0.65	0.76	0.23	0.27	0.31
$n=1000$	Initial	4.76	5.00	5.25	47.35	49.89	52.68	473.92	497.67	524.17
	RandSelect	NA	NA	NA	NA	NA	NA	NA	NA	NA
	BSA	0.25	0.27	0.28	2.41	2.54	2.66	24.10	25.31	26.66
	Cust BSA	0.22	0.23	0.24	0.22	0.23	0.24	0.22	0.23	0.25
Panel B: $m=10$										
$n=10$	Initial	2.21	5.30	10.31	22.73	53.64	106.44	225.39	516.61	995.99
	RandSelect	1.71	3.86	7.25	14.07	32.81	64.76	NA	NA	NA
	BSA	0.94	2.21	4.31	5.58	12.32	23.34	31.36	66.43	125.73
	Cust BSA	0.98	2.11	4.09	0.88	1.99	3.95	0.40	0.92	1.76
$n=100$	Initial	4.00	5.00	6.28	39.87	50.08	61.29	402.29	506.90	626.37
	RandSelect	3.51	4.47	5.43	34.84	43.65	54.98	NA	NA	NA
	BSA	1.02	1.27	1.53	9.40	11.63	14.00	60.63	75.13	90.63
	Cust BSA	0.91	1.12	1.39	1.43	1.79	2.18	0.88	1.10	1.34
$n=1000$	Initial	4.67	5.00	5.34	46.83	50.06	53.57	466.42	500.36	535.99
	RandSelect	NA	NA	NA	NA	NA	NA	NA	NA	NA
	BSA	0.68	0.73	0.79	6.51	6.94	7.41	64.13	68.42	73.04
	Cust BSA	0.61	0.66	0.70	0.77	0.82	0.87	0.89	0.95	1.02

Table 10: The same statistics (w.r.t. the variance) as in Table 1 for correlation parameter $\rho = -0.1$.

		$d=5$			$d=50$			$d=500$		
		min.	med.	max.	min.	med.	max.	min.	med.	max.
Panel A: $m=5$										
$n=10$	Initial	4.87	6.77	9.21	15.83	21.57	28.85	48.43	68.17	90.07
	RandSelect	3.91	5.43	7.31	11.07	15.69	20.89	NA	NA	NA
	BSA	2.75	3.73	4.97	5.16	7.07	9.29	10.09	13.94	18.30
	Cust BSA	2.64	3.57	4.61	1.91	2.60	3.38	0.98	1.34	1.72
$n=100$	Initial	9.74	11.16	12.93	30.48	35.22	40.33	97.54	112.33	129.44
	RandSelect	8.88	10.06	11.66	28.36	32.07	37.04	NA	NA	NA
	BSA	3.47	3.95	4.47	10.46	11.81	13.44	22.41	25.53	28.87
	Cust BSA	3.27	3.74	4.31	3.49	3.99	4.49	2.22	2.53	2.90
$n=1000$	Initial	13.34	14.50	15.84	42.21	45.83	50.12	132.84	145.12	158.96
	RandSelect	NA	NA	NA	NA	NA	NA	NA	NA	NA
	BSA	3.00	3.23	3.56	9.12	9.76	10.69	28.66	30.80	33.65
	Cust BSA	2.81	3.04	3.36	2.76	2.96	3.25	2.77	2.98	3.29
Panel B: $m=10$										
$n=10$	Initial	4.34	6.80	9.83	13.66	21.76	31.99	42.91	67.42	97.09
	RandSelect	3.76	5.84	8.28	10.92	16.99	24.80	NA	NA	NA
	BSA	2.81	4.42	6.38	6.73	10.39	14.84	16.20	23.99	34.31
	Cust BSA	2.82	4.28	6.27	2.66	4.19	6.13	1.80	2.83	4.05
$n=100$	Initial	9.35	11.15	13.60	29.52	35.03	42.77	93.96	111.74	136.77
	RandSelect	8.75	10.49	12.63	27.43	33.07	39.92	NA	NA	NA
	BSA	4.67	5.50	6.53	14.07	16.59	19.77	35.51	41.86	49.66
	Cust BSA	4.35	5.19	6.29	5.52	6.62	7.80	4.36	5.18	6.17
$n=1000$	Initial	12.97	14.43	16.48	41.09	45.30	52.03	129.72	144.74	164.58
	RandSelect	NA	NA	NA	NA	NA	NA	NA	NA	NA
	BSA	4.86	5.32	6.01	14.78	16.24	18.21	46.59	51.33	57.26
	Cust BSA	4.60	5.11	5.76	5.13	5.71	6.44	5.59	6.20	7.01

Table 11: The same statistics (w.r.t. the range) as in Table 3 for correlation parameter $\rho = -0.1$.

		$d=5$			$d=50$			$d=500$		
		min.	med.	max.	min.	med.	max.	min.	med.	max.
Panel A: $m=5$										
$n=10$	Initial	5.06	9.09	15.18	51.62	88.87	136.25	547.19	922.11	1450.98
	RandSelect	2.61	4.86	8.70	21.80	39.48	62.54	NA	NA	NA
	BSA	1.17	2.18	3.99	4.44	7.79	12.67	16.24	27.53	45.17
	Cust BSA	1.05	1.99	3.94	0.49	0.84	1.47	0.11	0.21	0.35
$n=100$	Initial	7.04	8.34	10.04	72.38	84.10	97.06	733.70	850.22	973.91
	RandSelect	5.53	6.51	7.89	55.01	64.69	76.12	NA	NA	NA
	BSA	0.64	0.77	0.96	6.15	7.36	8.81	30.44	35.73	41.59
	Cust BSA	0.58	0.69	0.85	0.61	0.72	0.86	0.22	0.26	0.31
$n=1000$	Initial	7.90	8.32	8.80	79.26	83.21	87.37	797.90	830.99	871.06
	RandSelect	NA	NA	NA	NA	NA	NA	NA	NA	NA
	BSA	0.30	0.32	0.37	3.06	3.21	3.37	30.28	31.87	33.43
	Cust BSA	0.25	0.27	0.31	0.23	0.24	0.25	0.22	0.23	0.25
Panel B: $m=10$										
$n=10$	Initial	4.05	8.91	17.95	46.34	89.67	166.32	430.88	879.16	1546.84
	RandSelect	2.39	5.12	10.77	19.99	45.00	86.83	NA	NA	NA
	BSA	1.17	2.79	6.41	6.81	14.82	28.38	35.29	81.32	150.91
	Cust BSA	1.13	2.58	5.76	1.03	2.32	4.53	0.34	0.75	1.48
$n=100$	Initial	6.82	8.46	11.36	68.29	83.06	101.14	692.10	836.94	1025.92
	RandSelect	5.35	6.79	8.97	54.02	66.77	81.93	NA	NA	NA
	BSA	1.20	1.51	2.40	11.48	14.41	17.72	72.37	90.90	111.36
	Cust BSA	1.08	1.35	2.16	1.56	1.97	2.44	0.72	0.90	1.10
$n=1000$	Initial	7.73	8.31	8.98	78.56	83.73	88.98	783.26	834.38	885.75
	RandSelect	NA	NA	NA	NA	NA	NA	NA	NA	NA
	BSA	0.79	0.86	0.97	7.77	8.33	8.90	78.52	83.99	89.80
	Cust BSA	0.70	0.76	0.86	0.74	0.80	0.85	0.76	0.81	0.86
Panel C: $m=20$										
$n=10$	Initial	3.20	8.45	20.51	35.92	90.60	185.47	349.88	879.22	1741.96
	RandSelect	NA	NA	NA	NA	NA	NA	NA	NA	NA
	BSA	1.10	3.25	9.42	7.44	21.89	48.22	53.95	142.52	307.40
	Cust BSA	1.15	3.26	8.87	1.94	5.38	12.32	0.75	2.04	4.58
$n=100$	Initial	6.45	8.52	11.98	65.47	83.50	109.58	655.12	830.22	1038.17
	RandSelect	NA	NA	NA	NA	NA	NA	NA	NA	NA
	BSA	1.69	2.21	3.41	15.84	21.06	29.14	115.95	153.40	198.10
	Cust BSA	1.55	2.07	3.29	3.39	4.50	6.56	1.77	2.37	3.07
$n=1000$	Initial	7.60	8.32	9.22	77.83	83.66	90.22	770.95	831.20	890.65
	RandSelect	NA	NA	NA	NA	NA	NA	NA	NA	NA
	BSA	1.39	1.53	1.75	13.60	14.81	16.08	135.44	147.49	160.22
	Cust BSA	1.28	1.41	1.61	1.88	2.04	2.21	2.09	2.27	2.46

Table 12: The same statistics (w.r.t. the variance) as in Table 1 with production costs drawn from a t_5 distribution.

		$d=5$			$d=50$			$d=500$		
		min.	med.	max.	min.	med.	max.	min.	med.	max.
Panel A: $m=5$										
$n=10$	Initial	6.53	8.87	12.20	20.67	27.80	35.60	67.63	91.11	115.50
	RandSelect	4.66	6.57	9.07	13.28	18.57	24.18	NA	NA	NA
	BSA	3.09	4.36	6.02	6.03	8.13	10.72	11.85	15.29	19.96
	Cust BSA	2.95	4.18	5.84	1.99	2.70	3.60	0.96	1.33	1.75
$n=100$	Initial	13.00	15.52	19.36	40.31	45.66	53.47	127.12	145.87	165.25
	RandSelect	11.65	13.62	16.74	35.19	40.23	46.74	NA	NA	NA
	BSA	3.82	4.56	6.02	11.47	13.23	15.43	25.71	28.96	32.80
	Cust BSA	3.68	4.45	5.70	3.67	4.22	4.92	2.19	2.50	2.87
$n=1000$	Initial	20.07	22.77	28.89	55.60	60.55	69.02	173.04	188.14	206.55
	RandSelect	NA	NA	NA	NA	NA	NA	NA	NA	NA
	BSA	3.92	4.84	7.70	10.28	11.32	13.15	32.25	34.63	37.43
	Cust BSA	3.65	4.58	7.02	2.82	3.04	3.33	2.78	3.00	3.25
Panel B: $m=10$										
$n=10$	Initial	5.71	8.89	13.37	19.46	27.97	39.94	59.14	87.52	121.93
	RandSelect	4.44	6.81	10.25	12.94	19.93	28.77	NA	NA	NA
	BSA	3.09	4.95	7.90	7.67	11.44	16.51	17.12	26.92	37.82
	Cust BSA	3.08	4.81	7.45	2.88	4.46	6.45	1.67	2.56	3.74
$n=100$	Initial	12.72	15.84	22.23	38.49	45.99	56.63	122.25	143.65	175.54
	RandSelect	11.15	13.85	19.07	34.58	41.08	50.95	NA	NA	NA
	BSA	5.19	6.38	10.04	15.63	18.71	22.87	39.27	46.04	55.98
	Cust BSA	4.91	6.08	9.63	5.85	6.96	8.58	3.92	4.63	5.58
$n=1000$	Initial	19.26	22.85	31.80	54.52	61.60	73.12	167.40	187.25	211.65
	RandSelect	NA	NA	NA	NA	NA	NA	NA	NA	NA
	BSA	5.83	7.16	11.89	16.24	18.08	21.84	51.41	56.45	63.18
	Cust BSA	5.56	6.92	11.40	5.10	5.65	6.41	5.11	5.65	6.35
Panel C: $m=20$										
$n=10$	Initial	5.14	8.70	14.54	16.61	28.06	43.17	53.13	87.61	129.55
	RandSelect	NA	NA	NA	NA	NA	NA	NA	NA	NA
	BSA	3.03	5.41	9.48	7.92	13.79	21.62	21.25	35.35	54.36
	Cust BSA	3.09	5.40	9.45	3.93	6.91	10.82	2.43	4.24	6.58
$n=100$	Initial	12.22	15.70	23.57	37.06	45.88	60.67	116.08	143.69	180.74
	RandSelect	NA	NA	NA	NA	NA	NA	NA	NA	NA
	BSA	6.01	7.77	12.41	18.20	22.56	30.36	49.47	60.84	76.32
	Cust BSA	5.90	7.48	12.16	8.46	10.51	14.14	6.18	7.57	9.61
$n=1000$	Initial	18.52	22.85	34.78	53.05	60.77	75.45	163.99	185.85	217.58
	RandSelect	NA	NA	NA	NA	NA	NA	NA	NA	NA
	BSA	7.51	9.28	16.69	21.55	24.33	30.59	67.44	76.22	88.15
	Cust BSA	7.20	8.98	16.19	8.05	9.08	10.78	8.47	9.56	11.03

Table 13: The same statistics (w.r.t. the range) as in Table 3 with production costs drawn from a t_5 distribution.

References

Bar-Shalom, Y. (1990): "Multitarget-multisensor tracking: advanced applications," *Norwood, MA, Artech House, 1990, 391 p.*

Bernard, C., L. Rüschendorf, S. Vanduffel, and R. Wang (2017): "Risk bounds for factor models," *Finance and Stochastics*, 21(3), 631–659.

Boudt, K., E. Jakobsons, and S. Vanduffel (2018): "Block rearranging elements within matrix columns to minimize the variability of the row sums," *4OR*, 16(1), 31–50.

Burkard, R. E. (1979): "Travelling salesman and assignment problems: A survey," in *Annals of Discrete Mathematics*, vol. 4, pp. 193–215. Elsevier.

Burkard, R. E., E. Çela, P. M. Pardalos, and L. S. Pitsoulis (1998): "The quadratic assignment problem," in *Handbook of Combinatorial Optimization. Vol. 3*, ed. by Du, Ding-Zhu and Pardalos, P.M., pp. 241–337. Boston: Kluwer Academic Publishers.

Camm, J. D., M. J. Magazine, G. G. Polak, and G. S. Zaric (2008): "Scheduling parallel assembly workstations to minimize a shared pool of labor," *IIE Transactions*, 40(8), 749–758.

Coffman Jr, E. G., and M. Yannakakis (1984): "Permuting elements within columns of a matrix in order to minimize maximum row sum," *Mathematics of Operations Research*, 9(3), 384–390.

Cornilly, D., G. Puccetti, L. Rüschendorf, and S. Vanduffel (2020): "Fair allocation of indivisible goods with minimum inequality or minimum envy criteria," *Available at SSRN 3512113*.

Embrechts, P., G. Puccetti, and L. Rüschendorf (2013): "Model uncertainty and VaR aggregation," *Journal of Banking & Finance*, 37(8), 2750–2764.

Gilbert, K. C., and R. B. Hofstra (1988): "Multidimensional assignment problems," *Decision Sciences*, 19(2), 306–321.

Haley, K. (1963): "The multi-index problem," *Operations Research*, 11(3), 368–379.

Hofert, M., A. Memartoluie, D. Saunders, and T. Wirjanto (2017): "Improved algorithms for computing worst Value-at-Risk," *Statistics & Risk Modeling*, 34(1-2), 13–31.

Hsu, W.-L. (1984): "Approximation algorithms for the assembly line crew scheduling problem," *Mathematics of Operations Research*, 9(3), 376–383.

Karapetyan, D., and G. Gutin (2011): "Local search heuristics for the multidimensional assignment problem," *Journal of Heuristics*, 17(3), 201–249.

Lee, C.-Y., and G. L. Vairaktarakis (1997): "Workforce Planning in Mixed Model Assembly Systems," *Operations Research*, 45(4), 553–567.

Loiola, E. M., N. M. M. de Abreu, P. O. Boaventura-Netto, P. Hahn, and T. Querido (2007): "A survey for the quadratic assignment problem," *European Journal of Operational Research*, 176(2), 657–690.

Pardalos, P. M., and L. S. Pitsoulis (2000): "Quadratic and multidimensional assignment problems," in *Nonlinear Optimization and Related Topics*, pp. 235–256. Springer.

Pesko, S. (2006): "The matrix permutation problem in interval arithmetic," *Journal of Information, Control and Information Systems*, 1(1).

Pesko, S., and R. Hajtmanek (2014): "Matrix permutation problem for fair workload distribution," in *Mathematical Methods in Economics*, ed. by T. Talášek, Olomouc, September 10–13.

Pierskalla, W. P. (1968): "Letter to the editor—the multidimensional assignment problem," *Operations Research*, 16(2), 422–431.

Poore, A., N. Rijavec, M. Liggins, and V. Vannicola (1993): "Data association problems posed as multidimensional assignment problems: problem formulation," in *Signal and Data Processing of Small Targets 1993*, vol. 1954, pp. 552–563. International Society for Optics and Photonics.

Puccetti, G. (2017): "An algorithm to approximate the optimal expected inner product of two vectors with given marginals," *Journal of Mathematical Analysis and Applications*, 451(1), 132–145.

Puccetti, G., and L. Rüschendorf (2012): "Computation of sharp bounds on the distribution of a function of dependent risks," *Journal of Computational and Applied Mathematics*, 236(7), 1833–1840.

Rüschendorf, L. (1983): "On the multidimensional assignment problem," *Methods of OR*, 47(107–113), 70.

Rüschendorf, L., and S. T. Rachev (1990): "A characterization of random variables with minimum L2-distance," *Journal of Multivariate Analysis*, 32(1), 48–54.

Spieksma, F. (2000): "Multi Index Assignment Problems: Complexity, Approximation, Applications," in *Nonlinear Assignment Problems, Algorithms and Application*. Kluwer.

Vairaktarakis, G. L., X. Cai, and C.-Y. Lee (2002): "Workforce planning in synchronous production systems," *European Journal of Operational Research*, 136(3), 551–572.