Computers and Operations Research 120 (2020) 104948

Contents lists available at ScienceDirect

Computers and Operations Research

journal homepage: www.elsevier.com/locate/cor

Heuristic methods to consecutive block minimization

Leonardo C.R. Soares[a,*], Jordi Alves Reinsma[b], Luis H.L. Nascimento[b], Marco A.M. Carvalho[b]

[a] Instituto Federal do Sudeste de Minas Gerais - Campus Manhuaçu, Manhuaçu Minas Gerais 36909-300, Brazil
[b] Universidade Federal de Ouro Preto - Campus Morro do Cruzeiro, Ouro Preto - Minas Gerais 35400-000, Brazil

ARTICLE INFO

Article history:
Received 1 August 2019
Revised 26 March 2020
Accepted 27 March 2020
Available online 4 April 2020

Keywords:
Combinatorial problems
Consecutive block minimization
Metaheuristic
Iterated local search
Exact algorithms

ABSTRACT

Many combinatorial problems addressed in the literature are modeled using binary matrices. It is often of interest to verify whether these matrices hold the consecutive ones property (C1P), which implies that there exists a permutation of the columns of the matrix such that all nonzero elements can be placed contiguously, forming a unique 1-block in every row. The minimization of the number of 1-blocks is approached by a well-known problem in the literature called consecutive block minimization (CBM), an $\mathcal{NP}$-hard problem. In this study, we propose a graph representation, a heuristic based on a classical algorithm in graph theory, the implementation of a metaheuristic for solving the CBM and the application of an exact method based on a reduction of the CBM to a particular version of the well-known traveling salesman problem. Computational experiments demonstrate that the proposed metaheuristic implementation is competitive, as it matches or improves the best known solution values for all benchmark instances available in the literature, except for a single instance. The proposed exact method reports, for the first time, optimal solutions for these benchmark instances. Consequently, the proposed methods outperform previous methods and become the new state-of-the-art for solving the CBM.

1. Introduction

Among the problems studied in discrete optimization, a series of them can be abstracted and modeled with binary matrices. This is common when dealing with equation systems, two-dimensional, or set-membership-related problems. Particularly, some of these are permutation problems, which arise in industrial environments. The gate matrix layout (GMLP) and the minimization of tool switching (SSP) are examples of problems with this feature (Linhares and Yanasse, 2002). Similarly, there are problems with applications in mathematics and algebra, such as the consecutive ones augmentation problem (Chakhlevitch et al., 2013) and consecutive ones problem C1P) (Atkins et al., 1998). Many binary matrix problems are linked to C1P, whose goal is to find a permutation of the columns of a matrix in which nonzero elements in every row appear consecutively (Fulkerson and Gross, 1965). This property, however, is most likely not found.

This study focuses on the consecutive block minimization (CBM), an $\mathcal{NP}$-hard problem (Garey and Johnson, 1979; Kou, 1977). It is defined as the problem of finding a permutation of the columns of a binary matrix, which minimizes the number of consecutive blocks. In any row of a binary matrix, a contiguous sequence of nonzero elements is called a *consecutive block* or *1-block*. The existence of any number of zeroes between two 1-blocks characterizes a *discontinuity*. The CBM is related to different areas of interest, although studied under different names. The problem of reducing the gap between elements occurs in archaeology as the *sequence dating problem* (Kendall, 1969), in microbiology as the *physical mapping problem* (Alizadeh et al., 1995), in the glass industry as the *minimization of discontinuities problem* (Dyson and Gregory, 1974; Madsen, 1979), in computational biology as a *Physical Mapping Problem* (Atkins and Middendorf, 1996; Christof et al., 1998) and in file organization without a specific name (Kou, 1977). The multidisciplinarity of the CBM is a motivational factor for its approach.

The above-mentioned industrial problems are intrinsically permutation problems, i.e., their solutions are determined by an ordering of the columns of the binary matrix. The evaluation of a solution is problem-specific; nonetheless, it considers the minimization of the number of discontinuities in the binary matrix. For example, in the GLMP, the columns represent gates and the rows represent nets that connect them. Nonzero elements indicate a transistor implemented at a specific gate and net. All transistors in the same net must be connected, and a solution to the problem consists of a permutation of columns that minimizes the length of the

* Corresponding author.
E-mail addresses: leonardo.soares@ifsudestemg.edu.br (L.C.R. Soares), jordiar@live.com (J.A. Reinsma), luis_ccm14@hotmail.com (L.H.L. Nascimento), mamc@ufop.edu.br (M.A.M. Carvalho).

https://doi.org/10.1016/j.cor.2020.104948
0305-0548/

nets. In the SSP, the columns represent jobs to be scheduled and the rows represent avaliable tools. Nonzero elements indicate that a job requires a specific tool to be loaded in the processing machine. The existence of a discontinuity indicates that a tool was replaced by another in the machine and later, loaded again, i.e., a tool switch, which requires the machine to be stopped. A solution to this problem consists in a job schedule that minimizes tool switches, thus, reducing machine idle time.

Formally, the CBM considers an $m \times n$ binary matrix $A = \{a_{ij}\}$, with $a_{ij} \in \{0, 1\}$, $i = \{1, 2, \ldots, m\}$, and $j = \{1, 2, \ldots, n\}$. The existence of a single 1-block in every row of A implies that it holds the *consecutive ones property* (C1P).

As an illustration, a binary matrix $A_{4,5}$ is shown as a hypothetical CBM instance. The first row has three 1-blocks caused by discontinuities at elements $a_{1,2}$ and $a_{1,4}$. The second row has three 1-blocks caused by discontinuities at $a_{2,2}$ and $a_{2,4}$. The third row has two 1-blocks, discontinued at $a_{3,3}$ and $a_{3,4}$. Finally, the fourth row has two 1-blocks discontinued at $a_{4,3}$. In total, this hypothetical instance has 10 1-blocks. The discontinuities are shown underlined on the mentioned matrix.

$$A = \begin{bmatrix} 1 & \underline{0} & 1 & \underline{0} & 1 \\ 1 & \underline{0} & 1 & \underline{0} & 1 \\ 1 & 1 & \underline{0} & \underline{0} & 1 \\ 0 & 1 & \underline{0} & 1 & 0 \end{bmatrix}$$

The CBM aims at minimizing the number of 1-blocks by generating a permutation of the matrix columns. Therefore, a solution for the instance represented by matrix A is a permutation π of its columns. The permutation matrix A^{π_1} shows a possible solution where the columns of A follow the order established by $\pi_1 = [2, 1, 3, 4, 5]$. The first two columns, shown in bold on A^{π_1}, had their positions exchanged. As a result, the number of 1-blocks in the first two rows is reduced to two and the third and fourth rows remain with two 1-blocks each. The total number of 1-blocks is reduced to eight.

$$A^{\pi_1} = \begin{bmatrix} \mathbf{0} & \mathbf{1} & 1 & 0 & 1 \\ \mathbf{0} & \mathbf{1} & 1 & 0 & 1 \\ \mathbf{1} & \mathbf{1} & 0 & 0 & 1 \\ \mathbf{1} & \mathbf{0} & 0 & 1 & 0 \end{bmatrix}$$

Although π_1 is a valid solution, it does not represent the optimal solution. For this example, the optimal solution is given by $\pi_2 = [3, 1, 5, 2, 4]$, with four 1-blocks. The permutation matrix A^{π_2} represents this solution.

$$A^{\pi_2} = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

The number of 1-blocks of a solution represented by a permutation matrix A^{π} is calculated using

$$Z^{\pi}_{CBM}(A^{\pi}) = \sum_{j=1}^{n} \sum_{i=1}^{m} A^{\pi}_{ji}(1 - A^{\pi}_{ji-1}) \tag{1}$$

proposed by Crama et al. (1994) for the job sequencing and tool switching problem. The equation calculates the number of inversions from zero to one. Although the number of inversions is greater than the number of 1-blocks, both are proportional and therefore such an equation can be used to measure the quality of the solution. For the 1-blocks starting in the first column, the matrix A^{π} is considered in (1) to have a fixed additional column zero with all values equal to zero, i.e., $A^{\pi}_{i0} = 0$.

As mentioned previously, CBM aims at minimizing the number of 1-blocks in a binary matrix. Therefore, it is possible to define its objective function as

$$\min_{\pi \in \Pi} \left\{ Z^{\pi}_{CBM}(A) \right\} \tag{2}$$

which aims at determining the permutation π of the columns of matrix A, among the set of all possible permutations Π that minimizes the number of 1-blocks.

Over the years, heuristic methods have been used to address the CBM, however, these methods were developed for specific contexts and were not evaluated or compared with other methods found in other studies. In this study, in addition to a new representation using graphs and a new heuristic, a metaheuristic and an exact method are employed for the first time to approach the problem. Benchmark instances available in the literature are used in the experiments and the results are compared with the current state-of-the-art.

The remainder of this article is organized as follows. The related work is summarized in Section 2 and our contributions to CBM are presented in Section 3. The computational experiments are reported in Section 4, and the conclusions follow in Section 5.

2. Related work

Problems involving 1-blocks in a binary matrix have been studied in the literature since the 1950s. In his thesis, Dom (2009) surveyed most of the literature about problems involving the C1P and 1-blocks in binary matrices. Kou (1977) and Garey and Johnson (1979) showed that the CBM belongs to the class of $\mathcal{NP}$-hard problems. Haddadi (2002), showed that even in a restricted case where the binary matrices have two 1-blocks per row, the problem is still $\mathcal{NP}$-hard.

Although CBM is a problem with wide practical applications and high theoretical relevance, the specific literature on CBM is very restricted and there is no approach using exact methods or metaheuristics, such as the proposals in this article. A common strategy to address CBM is modeling it as the traveling salesman problem (TSP) (Alizadeh et al., 1995; Dyson and Gregory, 1974; Haddadi and Layouni, 2008; Johnson et al., 2004; Kou, 1977; Madsen, 1988). In this strategy, TSP nodes represent each column of the binary matrix and some notion of "distance" between columns is employed. Then, the columns are sequenced using some TSP-specific heuristic, such as nearest neighbor, 2-opt or approximation, such as the Christofides algorithm (Christofides, 1976). To the best of our knowledge, only one study directly addresses the CBM. Although Haddadi et al. (2015) mentions that Johnson et al. (2004) was the only existing heuristic for the CBM at that time, it also employs the mentioned TSP modeling.

Dyson and Gregory (1974) addressed the cutting-stock problem (CSP) in the flat glass industry. A pattern allocation subproblem was modeled the CBM. A binary matrix indicated which cutting patterns produced each type of glass pieces, and the objective was to minimize the discontinuities in the production of the pieces, which could generate differences in their tones. The cutting stock problem was solved via linear programming and, the pattern allocation problem was modeled as the TSP. However, the computational resources did not allow the optimal solution, even for small instances. Thus, this modeling was discarded and a modified branch-and-bound algorithm was proposed.

The CBM is approached again as a pattern allocation subproblem by Madsen (1979). In this paper, the CBM was studied under the name of the minimization of discontinuities problem. The cutting patterns were obtained solving the knapsack problem and a heuristic originally applied to the matrix bandwidth problem was applied to reduce the production distance between pieces, and, consequently, to minimize the discontinuities. In another study in the same context, Madsen (1988) transformed the CBM instances

into TSP instances and generated solutions using a sub-optimal heuristic designed for the TSP.

Alizadeh et al. (1995) modeled a problem related to DNA sequencing as the CBM. The above mentioned TSP modeling was employed and solved heuristically. The distance matrix was created based on the Hamming distance between the columns of the CBM, which later proved to be a perfect mapping from the CBM to the TSP (Atkins et al., 1998).

Johnson et al. (2004) dealt with the CBM in the context of data compression. Original large boolean matrices matrices were consecutively partitioned into submatrices, which were tackled as smaller TSP instances. TSP heuristics were applied to each submatrix and solutions were merged to reconstitute the original matrices.

Haddadi and Layouni (2008) addressed the CBM to reduce the complexity of solving systems of linear equations. The authors presented a 1.5-approximation algorithm. For that, they utilized reductions between problems that do not modify the solution to the original problem. The binary matrix was addressed as an instance of the circular blocks minimization problem (Hsu and Connell, 2003), where the columns were considered in circular order. Consequently, the 1-blocks were also considered in this way. After that, the circular blocks minimization problem was transformed into an instance of the symmetric traveling salesman problem (Reinelt, 1994). The distance matrix considered the Hamming distance between the columns of the original problem, thus characterizing the Hamming traveling salesman problem (HTSP) (Ernvall et al., 1985). After the transformations, the authors employed the Christofides algorithm (Christofides, 1976), the approximation factor of which is 1.5, to find a solution for the traveling salesman problem in the transformed instance. The transformation of CBM to HTSP is illustrated in Subsection 3.4.

Haddadi et al. (2015), approaching the CBM in the same context of reducing the dimensions of linear systems, proposed a heuristic method based on two local search procedures. The first local search exchanges the positions of two non-adjacent columns of the solution. The neighborhood of size $\mathcal{O}(n^2)$ is explored to consider all possible permutations. When an improvement is achieved, the solution is updated and the process continues with the new permutation. If no such best permutation exists, the procedure ends. The second local search removes a column from its original position and reinserts it at a destination position. Again, the neighborhood has size $\mathcal{O}(n^2)$. Each column is analyzed for reinsertion in the search for a permutation that minimizes the number of 1-blocks. When an improvement is achieved, the solution is updated and the process continues with the new permutation. The procedure ends when no improvement is possible. The algorithm proposed by Haddadi et al. (2015), from a randomly generated initial solution, applies the reinsertion local search until a local optimum is found. Then, the exchange local search is applied from the that solution until a local optimum is found in the second neighborhood. There are no strategies to escape from local optima in the search for a global optimum or better solutions.

In addition to local search procedures, Haddadi et al. (2015) proposed a fast evaluation function, or δ-evaluation, that allows the effect of the local search moves to be evaluated without the need to re-evaluate the solution completely. Although the complete evaluation of a solution requires $\Theta(mn)$ complexity, the proposed δ-evaluation complexity is bounded by $\mathcal{O}(m)$. The computational experiments considered five real-world instances of a German railway company and 45 new artificial instances randomly generated with different characteristics. Currently, this is the only set of benchmark instances available in the literature, and the results of Haddadi et al. constitute the state-of-the-art for solving the CBM and its results are considered in the experiments reported in Section 4.

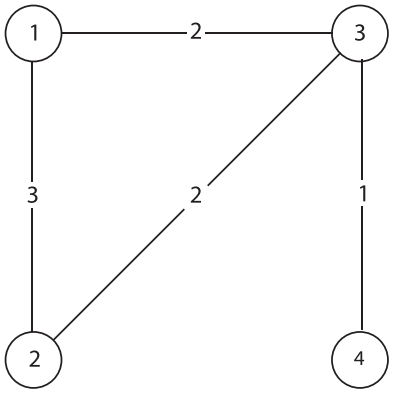


Fig. 1. CBM graph.

3. Methods

This section describes in details the following new contributions proposed to address the CBM: (i) a graph representation; (ii) a graph-based heuristic; (iii) the use of a tailored local search procedure; (iv) an implementation of an iterated local search (ILS) metaheuristic; and (v) the implementation of an exact method for CBM.

3.1. A new graph representation

In a CBM graph, nodes represent the rows of the input matrix and edges connect the nonzero elements of each column, whose weights are determined by the frequency of such an event. The main idea is to escape the pattern of approaching the problem from the columns perspective. Instead, we analyze the rows for information that can be used as heuristic rules. Unlike traditional approaches, the occurrence of nonzero elements is considered per row and not per columns. This graph representation resembles the utilized in Carvalho and Soma (2015), Paiva and Carvalho (2017), and Lima and Carvalho (2017) and the *intersection graph* introduced by Fulkerson and Gross (Fulkerson and Gross, 1965), however, the edge weight is specific for the CBM.

As an example, the binary matrix A represents an instance of CBM and Fig. 1 presents the correspondent CBM graph representation with four nodes. The first column of A has nonzero elements at rows 1, 2, and 3, thus, the edges {1, 2}, {1, 3}, and {2, 3} are added to the graph, with weight one. The second column has nonzero elements at rows 3 and 4, thus, an edge {3, 4} is also added to the graph, with weight one. The third column has nonzero elements at rows 1 and 2. The edge {1, 2} already exists in the graph, thus, its weight is updated to two. The fourth column has nonzero elements only at row 4 and no edge is added to the graph. The fifth column has nonzero elements at rows 1, 2, and 3. All of the corresponding nodes are already connected by edges, thus, edge {1, 2} has its weight updated to three, edge {1, 3} has its weight updated to two, and edge {2, 3} has its weight updated to two.

$$A = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \end{bmatrix}$$

3.2. A graph-based heuristic

The proposed representation in graphs offers a different perspective of CBM. Instead of the traditional column permutation perspective, it offers a perspective on the relationship of the rows,

which provides useful insights for solving the CBM. Therefore, we propose a greedy heuristic based on the traversal of the new graph representation. The rationale behind it is to obtain a row sequencing and, subsequently, use this information to obtain the desired solution of the CBM, a column permutation. This strategy has been used successfully in different problems that consider binary matrices as their abstractions (Carvalho and Soma, 2015; Lima and Carvalho, 2017; Paiva and Carvalho, 2017). In this article, we use breadth-first search (BFS) as a traversal algorithm, with some adaptations.

Given an initial node, the BFS explores all of its neighbors. After that, for each neighboring node, the BFS explores its entire neighborhood. Each node is explored once. This process is repeated until the entire graph is explored. If there is more than one component in the graph, the BFS examines an entire component, and then a node present in another component is selected to restart the search. Therefore, decomposable instances are processed naturally. The BFS returns a list ϕ with the order in which all the nodes present in the graph were explored.

The first modification made in BFS is the choice of the initial node. Our implementation selects the node with the lowest degree as the initial node. The rationale is that such node, with fewer related rows, should not be scheduled in the middle of a solution, as strongly related rows must be scheduled contiguously. In the case of ties, these are addressed by selecting nodes in lexicographic order. The second modification regards the order of exploration of nodes. In a neighborhood, a node has priority proportional to the weight of the edge that connects it to the node whose neighborhood is being explored. Again, ties are addressed by selecting nodes in lexicographic order. The purpose of this greedy criterion is to sequence the elements that often appear together and later use this information in the permutation of the columns.

The pseudocode for the adapted BFS is presented in Algorithm 1 . A CBM graph $G = (V, E)$ is given as an input

Algorithm 1: BFS.

```
   input  : CBM graph G = (V, E);
   output: list of nodes φ;
1  create an empty queue Q;φ ← ∅;while V ≠ ∅ do
2  |  i ← the lowest degree node in V;insert i at the end of
   |  φ;Q ← i;V \ i;while Q ≠ ∅ do
3  |  |  v ← Q.dequeue();foreach w ∈ N(v) do
4  |  |  |  V \ w;Q ← w;insert w at the end of φ;
5  return φ;
```

parameter. An empty queue is created to guide the search and the list ϕ is initialized as empty (lines 1 and 2). The main loop (lines 3–13) ensures that all nodes in G are explored. The node i with the lowest degree is selected as the initial one (line 4), added to list ϕ (line 5), to the queue (line 6), and removed from the graph (line 7). The second loop (lines 8–15) ensures the exploration of each component of G. The node v in the front of the queue is removed (line 9) and each node in the neighborhood of v, denoted by $N(v)$, is removed from the graph and added to Q and ϕ (lines 11–13). After exploring all nodes, the algorithm ends and the list ϕ is returned (line 17). All nodes and edges are explored, therefore, the BFS runs in time $\mathcal{O}(|V| + |E|)$, using an adjacency list representation.

Fig. 2 illustrates the application of the adapted BFS to a CBM graph, presented previously in Fig. 1. Nodes can be unexplored or explored, indicated by white and gray, respectively. Initially, as shown in (a), all nodes are unexplored and, thus, ϕ is empty. In (b), node 4 is selected because it has the lowest degree among all nodes. This node is added to ϕ and its neighborhood is explored. Its only neighbor, node 3, is explored and added to ϕ, as shown in (c). Node 3 has two neighbors, nodes 1 and 2. Both have the same degree and the tie is solved by lexicographic order. Node 1 is added to ϕ (d) and, then, node 2 is also added to ϕ (e). As there are no more nodes to be explored, the BFS ends and the list $\phi = [4, 3, 1, 2]$ is returned.

The BFS generates an ordered sequence of rows of the input matrix. However, a feasible solution to the CBM consists of a permutation π of columns of the matrix, therefore, it is necessary to generate such permutation from ϕ. Becceneri et al. (2004) presented a greedy method for this purpose in the context of the minimization of open stacks, a problem related to CBM. The method traverses the list of rows and sequences columns. A column is inserted into the solution π once the rows of its nonzero elements are matched by the traversed rows in ϕ. If more than one column meets this criterion, the lexicographic order is adopted.

Algorithm 2 presents the pseudocode to the greedy column

Algorithm 2: Greedy column sequencing.

```
   input  : list of rows φ, binary matrix A;
   output: permutation of columns π;
1  π ← ∅;R ← ∅;foreach i ∈ φ do
2  |  R ← R ∪ i;foreach column j ∈ A do
3  |  |  if c(j) ⊆ R and j ∉ π then
4  |  |  |  add j to the end of π;
5  return π;
```

sequencing. The list of rows ϕ is given as an input parameter as well as the binary matrix A_{mn} of CBM. Initially, the permutation of columns π and the auxiliary set of rows R are empty (lines 1 and 2). The main loop (lines 3–7) adds each element of ϕ to R (line 4). Then a second loop (lines 5–7) checks whether any subset of R represents the row indices of nonzero elements of any column in A. An auxiliary function $c(j)$ returns the row indices of the nonzero elements of column j. If the criterion is met (line 6) and if the column has not yet been added to π, the column is added to the end of it (line 7). After all columns are inserted into the solution, the algorithm ends and the permutation π is returned (line 11). The greedy column sequencing requires the examination of the entire binary matrix. Therefore, the procedure time complexity is bounded by $\mathcal{O}(mn)$.

We illustrate the application of the Algorithm 2 on the list ϕ returned in the BFS application example. Analyzing matrix A we have $c(1) = \{1, 2, 3\}$, $c(2) = \{3, 4\}$, $c(3) = \{1, 2\}$, $c(4) = \{4\}$, and $c(5) = \{1, 2, 3\}$. Traversing $\phi = [4, 3, 1, 2]$, column 4 is sequenced after row 4; column 2 is sequenced after row 3; no column is sequenced after row 1, and columns 1, 3, and 5 are sequenced in lexicographic order after row 2. The permutation $\pi = [4, 2, 1, 3, 5]$ is the solution obtained by the greedy column sequencing and matrix A^{π} is obtained using permutation π on matrix A. Comparing A and A^{π}, the number of 1-blocks has reduced from 10 to 5.

$$A^{\pi} = \begin{bmatrix} 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 \end{bmatrix}$$

3.3. ILS

Different metaheuristic methods search the solution space by coordinating procedures of *intensification* and *diversification* until a stop criterion is met. The intensification step employs *local search* procedures, which systematically apply a specific *move*, i.e., a specific type of modification, on a solution s, generating new solution s'. The region of the search space reachable by the applications

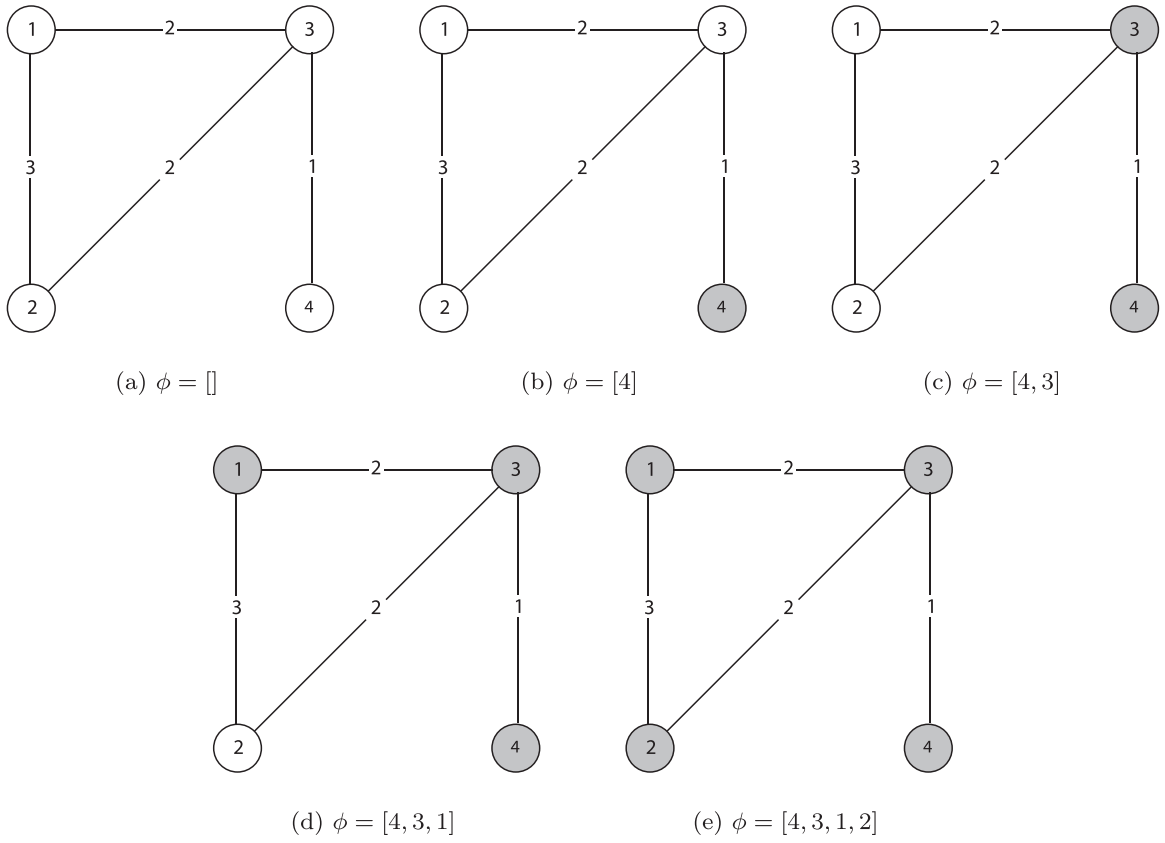


Fig. 2. BFS applied to a CBM graph.

of a particular move is called the *neighborhood*, and a solution s' is called the *neighbor solution* of s. If s' is the best solution on a given neighborhood, it is considered a *local optimum*. However, a *global optimum*, i.e., a local optimum common to all possible neighborhoods, may lie at another region of the search space. To diversify the solutions found, a mechanism of *perturbation* is employed. Such a mechanism causes certain levels of disorder in the solution to reach other regions of the solution space previously unreachable by a local search. The balance between intensification and diversification is an approach to keep the search from becoming stuck at local optima.

The ILS metaheuristic (Lourenço et al., 2019) performs a biased sampling of the solution space. Given an initial solution s to a specific optimization problem, the ILS consists of local searching on s and then alternately applying perturbation and local search procedures, generating a new solution s' that may be used as the starting point for the next iteration of the method. Although conceptually simple, the ILS has led to state-of-the-art algorithms for many computationally hard problems (Lourenço et al., 2019). In addition, its features match our intended approach to the CBM: the constructive heuristic could be used to generate the initial solution and local search procedures specific to the problem could be applied during the intensification step. Additionally, the perturbation step could avoid the search being trapped in (not global) local optima.

In this study, an initial solution for the CBM was constructed by employing the new representation using CBM graphs and the graph-based heuristic described in the previous sections. Considering the characteristics of CBM, our ILS implementation employs the 2-opt method proposed by Croes (1958) and the local search by the grouping of 1-blocks proposed by Paiva and Carvalho (2017) in the intensification step. In addition, the 2-swap method is employed in the diversification step. In the face of the trade-off between running time and solution convergence, we decided to restrict the application of the perturbation to a percentage α of the solution size. The stopping criteria and the α parameter are defined in the preliminary experiments described in Section 4.

Algorithm 3 presents the pseudocode for the proposed ILS. The

Algorithm 3: Iterated local search.

```
   input  : α;
   output: solution π;
1  π ← initialSolution();while stopping criteria not met do
2  |  π′ ← swap α random columns of π;apply the 2-opt local
   |  search to π′;apply the 1-blocks grouping local search to
   |  π′;if π′ is better than π then
3  |  |  π ← π′;
4  return π;
```

parameter α is provided as an input parameter. The initial solution π is generated using BFS on the CBM graph and greedy column sequencing. The main loop (lines 2–9) performs the ILS while any of the stopping criteria are not met. The current solution π undergoes a perturbation. All possible pairs of column indices are shuffled and $\alpha/2$ of such pairs are selected for swapping, generating a α-neighboring solution π' (line 3). Then, the local search is performed by applying sequentially the 2-opt (line 4) and 1-blocks grouping (line 5) methods. At this point, π' is a local optimum solution to both neighborhoods considered. If its solution value is better than the solution value of π (line 6), then π' is accepted as the new current solution (line 7) and will be the starting point to the next ILS iteration. Otherwise, π' is discarded and π remains as the current solution. Finally, the algorithm returns the solution π.

The intensification and diversification components of the ILS, as well as a fast evaluation procedure, are described next.

3.3.1. 2-Swap method

A move in the well-known 2-swap method consists of exchanging two elements of a given solution. All $\mathcal{O}(n^2)$ possible pairs of column indices are generated and shuffled. Then, $\frac{\alpha}{2}$ of such pairs are selected for swapping their positions in the solution.

As an example, let us consider a solution $\pi = [1, 2, 3, 4, 5]$ illustrated by matrix A^{π}, with five 1-blocks. A pair of elements of the solution is selected randomly for exchange: in this specific case, the elements 2 and 4, bolded in A^{π}. Then, the 2-swap method exchanges the selected elements generating the permutation $\pi' = [1, 4, 3, 2, 5]$, as illustrated by matrix $A^{\pi'}$ with seven 1-blocks.

$$A^{\pi} = \begin{bmatrix} 0 & \mathbf{0} & 1 & \mathbf{1} & 1 \\ 0 & \mathbf{0} & 1 & \mathbf{1} & 1 \\ 0 & \mathbf{1} & 1 & \mathbf{0} & 1 \\ 1 & \mathbf{1} & 0 & \mathbf{0} & 0 \end{bmatrix} \quad A^{\pi'} = \begin{bmatrix} 0 & \mathbf{1} & 1 & \mathbf{0} & 1 \\ 0 & \mathbf{1} & 1 & \mathbf{0} & 1 \\ 0 & \mathbf{0} & 1 & \mathbf{1} & 1 \\ 1 & \mathbf{0} & 0 & \mathbf{1} & 0 \end{bmatrix}$$

3.3.2. 2-Opt method

A move in the 2-opt method, originally proposed by Croes (1958) for the traveling salesman problem, consists of reversing the positions of the elements of a permutation within a given interval. All $\mathcal{O}(n^2)$ possible pairs of columns indices are generated and shuffled. Then, the move is evaluated for every possible pair of indices in the solution, sequentially. In the case of a solution value improvement, the move is kept and the whole procedure restarts. Otherwise, the move is discarded and the next pair is evaluated. The procedure stops when improving moves do not exist.

As an example, consider the solution $\pi = [1, 2, 3, 4, 5]$ and the closed interval (2, 5). Matrix A^{π} represents the initial solution and has five 1-blocks. The columns of the given interval are bolded. A move of the 2-opt method reverses the elements of the interval generating the solution $\pi' = [1, 5, 4, 3, 2]$ shown in matrix $A^{\pi'}$, with six 1-blocks.

$$A^{\pi} = \begin{bmatrix} 0 & \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{1} \\ 0 & \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{1} \\ 0 & \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{1} \\ 1 & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix} \quad A^{\pi'} = \begin{bmatrix} 0 & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{0} \\ 0 & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{0} \\ 0 & \mathbf{1} & \mathbf{0} & \mathbf{1} & \mathbf{1} \\ 1 & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} \end{bmatrix}$$

3.3.3. 1-Blocks grouping

Paiva and Carvalho (2017) approached the job sequencing and tool switching problem, which is found in the context of flexible manufacturing systems. In such a context, they proposed the 1-blocks grouping local search with the aim of reducing the number of discontinuities in a binary matrix. For that, the local search analyzes the binary matrix row by row, in random order, searching for two or more 1-blocks. For each pair of consecutive 1-blocks, the procedure tries to move the columns of the first 1-block, one by one, to the left and to the right of the second 1-block, in an attempt to group them. Each move is analyzed, and the one that results in the best solution value is performed. All non-improving moves are discarded, and the procedure stops when all rows have been considered. The analysis of the 1-blocks is performed in time $\mathcal{O}(mn)$. The evaluation of a possible grouping can be performed in $\mathcal{O}(m)$ time, as described next. Therefore, the complexity of this local search is bounded by $\mathcal{O}(m^2 n)$.

A move of this local search is illustrated by the matrices (a)-(d) below. In (a), we have a binary matrix with five 1-blocks. In (b), the second row is selected randomly for analysis. It has two 1-blocks, one on the first column and another in the third column. The local search evaluates moving the column of the first 1-block to the second and fourth positions, i.e., to the left and to right of the 1-block on the third position. The move to the left (c) decreases the total of 1-blocks to four. The move to the right (d), increases the number of 1-blocks to six. Therefore, the move to the left results in the best solution value and is performed.

(a) $\begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}$ (b) $\begin{bmatrix} 0 & 1 & 0 & 0 \\ \mathbf{1} & \mathbf{0} & \mathbf{1} & \mathbf{0} \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}$

(c) $\begin{bmatrix} 1 & \mathbf{0} & 0 & 0 \\ 0 & \mathbf{1} & 1 & 0 \\ 0 & \mathbf{0} & 1 & 1 \\ 0 & \mathbf{0} & 1 & 1 \end{bmatrix}$ (d) $\begin{bmatrix} 1 & 0 & \mathbf{0} & 0 \\ 0 & 1 & \mathbf{1} & 0 \\ 0 & 1 & \mathbf{0} & 1 \\ 0 & 1 & \mathbf{0} & 1 \end{bmatrix}$

3.3.4. Fast evaluation procedure

The complete evaluation of a solution to the CBM has complexity bounded by $\Theta(mn)$, where m is the number of rows and n is the number of columns of the input matrix. Performing such evaluation for every possible move in the local search at every ILS iteration is computationally costly. To reduce the computational effort and consequently improve the local search performance, we employed the δ-evaluation presented by Haddadi et al. (2015) for CBM. Instead of evaluating the whole matrix, this fast evaluation procedure analyzes only the impact of a column changing its position in the solution, with complexity bounded by $\mathcal{O}(m)$.

Specifically, the number of 1-blocks created by the insertion of a column j between two adjacent columns i and k of a binary matrix B can be calculated by the number of occurrences of the patterns 101 and 010 in each row r of the matrix, given by

$$\delta(i, j, k) = \sum_{r=1}^{m} \left(B_j^r - B_i^r \times B_j^r - B_j^r \times B_k^r + B_i^r \times B_k^r \right) \tag{3}$$

In our ILS implementation, the δ-evaluation is used in both local search procedures, given that they produce neighborhoods of size $O(n^2)$ that are fully explored.

3.4. An exact method for the consecutive block minimization

The CBM reduction to TSP was formally proven by Haddadi and Layouni (2008). The main steps of the reduction are illustrated next. Given a binary $m \times n$ matrix A, there are $\frac{n!}{2}$ unique CBM solutions owing to the reflexivity of permutations of the n columns of the matrix. The TSP, given n nodes to visit, has $\frac{(n-1)!}{2}$ unique solutions owing to the reflexivity and circular properties. Converting the CBM problem such that it has the same number of solutions as the TSP is the first part of the reduction.

Let us introduce the circular blocks minimization problem (CIR). Similar to CBM, CIR aims to minimize the number of consecutive blocks of a binary matrix. However, the first and last columns of that matrix are considered adjacent to each other, thereby granting CIR the circular property.

Next, we show how a CBM instance can be transformed to a CIR instance while preserving its optimality. Without loss of generality, matrix A will be used as an example. A column permutation $\pi_1 = [a, b, c, d, e, f]$ generates matrix A^{π_1}, which is an optimal solution for CIR, but not for CBM. An optimal CBM solution is given by $\pi_2 = [d, b, c, a, f, e]$ and represented as A^{π_2}.

Now, by inserting a column '0' of zeros in A, and solving the CIR again for the resulting matrix B, we may find the optimal solution

$$A^{\pi_1} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{array} \begin{array}{c} \begin{array}{cccccc} a & b & c & d & e & f \end{array} \\ \begin{pmatrix} 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 \end{pmatrix} \end{array} \qquad A^{\pi_2} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{array} \begin{array}{c} \begin{array}{cccccc} d & b & c & a & f & e \end{array} \\ \begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 \end{pmatrix} \end{array}$$

$$B = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{array} \begin{array}{c} \begin{array}{ccccccc} 0 & a & b & c & d & e & f \end{array} \\ \begin{pmatrix} 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 \end{pmatrix} \end{array} \quad D_h = \begin{array}{c} \\ 0 \\ a \\ b \\ c \\ d \\ e \\ f \end{array} \begin{array}{c} \begin{array}{ccccccc} 0 & a & b & c & d & e & f \end{array} \\ \begin{pmatrix} 0 & 3 & 2 & 2 & 2 & 2 & 2 \\ 3 & 0 & 3 & 3 & 3 & 5 & 3 \\ 2 & 3 & 0 & 4 & 4 & 2 & 4 \\ 2 & 3 & 4 & 0 & 4 & 4 & 2 \\ 2 & 3 & 4 & 4 & 0 & 2 & 2 \\ 2 & 5 & 2 & 4 & 2 & 0 & 4 \\ 2 & 3 & 4 & 2 & 2 & 4 & 0 \end{pmatrix} \end{array}$$

$\pi_3 = [a, b, c, 0, d, e, f]$. Considering the circular property of the solution, it may be equivalently represented as $\pi_3 = [0, d, e, f, a, b, c]$. As the first column in the permutation is filled with zeros, there is no occurrence of any circular block. Consequently, π_3 is simultaneously optimal for CIR and CBM. If the zero-column is removed, the same CBM solution is found.

To reduce the CIR-transformed instance to a TSP instance, "distances" between pairs of columns of the matrix need to be defined to represent the distance of one node to another in a TSP tour. Different authors show that Hamming distance between columns results in perfect mapping of the solution spaces (Alizadeh et al., 1995; Atkins et al., 1998), as it concerns the number of element mismatches between columns. In fact, the total tour distance of a TSP-transformed CIR instance using Hamming distance is equal to double the number of circular blocks of the CIR instance (Haddadi and Layouni, 2008). An example of Hamming distance matrix D_h of binary matrix B is mentioned below.

After this reduction, the Applegate et al. ((2006), the state-of-the-art method for solving the symmetric TSP optimally, was used to solve the resulting TSP instance. With the TSP solution generated, the CBM matrix has its columns permuted according to the tour with the zero-column being shifted to the first position of the tour and subsequently removed. Therefore, a solution for CBM is generated.

4. Computational experiments

A series of computational experiments were performed to tune the ILS parameters, to compare the obtained results by ILS and the exact method with the state-of-the-art method and to measure the quality of the best known solutions against the optimal solutions or new upper bounds reported by the exact method. The next sections describe the computational environments used, the benchmark instances considered and report the experimental results. Detailed results for each experiment can be found in the supplementary material.

Three distinct computational environments were employed in this study.For the constructive heuristic and ILS, the computational environment adopted for the computational experiments consisted of an Intel Core i7-8700 3.20 GHz with 16 GB of RAM under the Ubuntu 18.04 operating system. These methods were implemented using the C++ language and compiled using g++ 4.4.1 and the -O3 optimization option.

The CBM-TSP reduction was coded in C++ and run on an Intel Core i7-4790 3.60 GHz four-core processor with 16 GB RAM. The Concorde solver used was implemented by Hans Mittelmann and made available at the NEOS Server for Optimization (Mittelmann, (2016) using the CPLEX linear programming solver.

Table 1
Artificial instances.

Group	*A*	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*
Density (%)	2	5	10	2	5	10	2	5	10
Dimensions	100 × 200			100 × 500			100 × 1000		

This heterogeneous environment comprises of an Intel i7-990X 3.47 GHz six-core processor with a 24 GB RAM, an Intel i7-7700K 4.20 GHz four-core processor with a 32 GB RAM, and an Intel Xeon X5690 3.47 GHz six-core processor with a 32 GB RAM. A time limit of 4 h was imposed on the solution of each instance. In case this limit was reached, an upper bound was reported.

The 45 artificial instances and 5 real-world instances used in the experiments of Haddadi et al. (2015) are considered. Each column of the artificial instances contains at least one nonzero element, and each row contains at least two 1-blocks. As listed in Table 1, the artificial instances are divided into nine groups (*A*–*I*) of five instances each. The real-world instances, whose dimensions are listed in Table 2, arise from a station location problem posed by a German railway company.

4.1. Parameter tuning

As mentioned in Section 3.3, our ILS has parameters concerning the stopping criteria and the percentage α of perturbation. A common stopping criterion is to set a limit of *max_iter* iterations for the ILS main loop. However, we learned that only a single static value was not enough, given the difference in the dimensions and densities of the instances. Even a small value of *max_iter* could lead to a prohibitive running time for large and dense instances, although it might serve well for small and sparse instances. Owing to that and considering the trade-off between running time and solution quality, we considered a dynamic limit of *max_time* seconds of runtime as a second stopping criterion.

To define the parameters, we employed the *irace* offline method for automatic configuration of optimization algorithms (López-Ibáñez et al., 2016). Given a set of instances for an optimization problem and a set of candidate values, the irace determines the best combination of values for the parameters. In this experiment, the training set is a representative subset of the set of instances, consisting of 5 instances (10% of the total) were randomly selected among all subgroups of the instances. In preliminary experiments we considered different values for the parameters. The ILS was not sensitive to small changes in the parameter values, then, the values were categorized, omitting intervals and narrowed down the options to the values presented in Table 3, which also presents the values selected by irace.

4.2. Comparison of ILS with the state-of-the-art

After tuning the ILS parameters, the results obtained by the method were compared with the results obtained by Haddadi et al. (2015). An in-depth comparison of results was not possible, because Haddadi et al. (2015) only reported average solution values. Nonetheless, detailed ILS results can be found in the supplementary material. A fair comparison of the running times was also not possible, as the methods were run in different computational environments. However, the heuristic of Haddadi et al. (2015) is simple and consists of a single application of two best improvement search procedures and, therefore, its running time is often negligible.

The following tables present the state-of-the-art results and average data from 20 independent runs of the ILS for each individual instance. For the heuristic proposed by Haddadi et al. (2015), the originally reported average solution values are presented (*BKS*).

Table 2
Real-world instances.

Instance	$RCOV_{1km}$	$RCOV_{2km}$	$RCOV_{3km}$	$RCOV_{5km}$	$RCOV_{10km}$
Dimensions	757 × 707	1196 × 889	1419 × 886	1123 × 593	275 × 165

Table 3
Parameter tuning.

Parameter	Options	Selected Value
α (%)	{5, 10, 15, 20, 25, 30, 35, 40, 45, 50}	10
max_iter	{50, 100, 150, 200, 250}	150
max_time	$\{\frac{n}{3}, \frac{n}{2}, n, 2 \times n\}$	$\frac{n}{2}$

For the proposed ILS implementation, the tables present the initial solution given by the constructive heuristic (I), the best solution value found (S^*), the average solution value (S), the standard deviation of S (σ), the average running time in seconds (T), and the gap (or percentage distance) from the best known solution, calculated as $100 \times \frac{\text{ILS solution - BKS}}{\text{BKS}}$. Note that the *ILS solution* may be S^* (column gap* (%)) or S (column gap (%)). Bold values identify solutions that reached the best values. Table 4 lists the results for the artificial instances.

According to the reported data, the proposed ILS generates new best average solutions for all groups of artificial instances, outperforming the state-of-the-art results. The overall average gap is - 10.26%, ranging from - 5.93% to - 15.25%. Should we consider only the average of the best solution values the average gap is - 10.68%, ranging from - 6.18% to - 15.87%. The smallest gap values regard the largest instances. The average standard deviation of 3.56 demonstrates the consistency of the ILS in generating high-quality solutions with low variation over independent runs. The limit on the running time was the predominant stopping criterion, as the mean values approximate $\frac{n}{2}$. For the larger instances (from groups *G*, *H*, and *I*), the running time is approximately 8.3 min, a good trade-off considering the solution quality.

The new constructive heuristic produced solutions in under 0.01 s for each instance with an average gap of 20.07%, ranging from 13.75% to 27.22%, regarding the solutions of Haddadi et al. (2015). On average, the ILS required 10.45 iterations to improve the initial solutions in 35.92%.

Additionally, a second version of the ILS using random initial solutions (ILS_r) was evaluated to accurately assess its convergence. Random solutions were produced in under 0.01 s for each instance. The average gap of these solutions, regarding the solutions of Haddadi et al. (2015), was 54.59%, with a maximum of 95.50%. On average, the ILS_r required 8.09 iterations to improve the initial solutions in 61.49%. The initial results generated by the new constructive heuristic outperform the random initial solutions with an average gap of 29.39%, up to 71.87%. Notwithstanding the inferior quality of the random initial solutions, the ILS has converged to solutions with the same quality of its previous version without an increase in the number of iterations, demonstrating the robustness of the proposed method.

Table 5 presents the average results for the five real instances, following the same standards of the previous table.

Except for the instance $RCOV_{2KM}$ (gap of 0.33%), new best solution values were established. Considering the other instances the percentage distances are shorter than those of the previous set of instances, demonstrating the competitiveness between the compared methods. The only instance with a proven optimal solution is $RCOV_{1KM}$, which reaches the trivial lower bound on the solution value of m, i.e., one 1-block in each matrix row. All ILS results reported are very close to this bound, with 2.11% being the largest percentage distance from it, on instance $RCOV_{3KM}$.

The initial solution values matched the best known values on three instances ($RCOV_{1KM}$, $RCOV_{2KM}$, and $RCOV_{3KM}$). For the other two instances ($RCOV_{4KM}$ and $RCOV_{5KM}$), the gap values between the initial solution values and Haddadi et al. (2015) are 3.15% and 2.5%, respectively. On average, the ILS required 7.85 iterations to improve the initial solutions by 3.47% and 3.99%, respectively, for instances $RCOV_{4KM}$ and $RCOV_{5KM}$. A statistical analysis was performed to compare the initial solutions achieved by the constructive heuristic and the method of Haddadi et al. (2015). The Shapiro and Wilk (1965) normality test, with a confidence interval of 95%, confirmed the null hypothesis that the results compared, constructive heuristic ($W = 0.92275,\ p = 0.5478$) and the method of Haddadi et al. (2015) ($W = 0.94085,\ p = .6719$), could be modeled according to a normal distribution. The *Student's t-test* (Student, 1908) was applied to verify whether there is a significant difference between the methods compared and indicated that there is no a significant difference between the results compared ($t = 0.88192,\ df = 4,\ p = 0.7862$) for a significance level of 0.05.

The independent runs yielded in standard deviation zero on four instances, and 0.4% deviation on the other, strengthening the indices claim of the ILS. Again, the running times approximate $\frac{n}{2}$, meaning that the time limit is the predominant stopping criterion. The maximum running time is 7.4 min for the larger instances.

Statistical tests compared the proposed ILS with the method of Haddadi et al. (2015). The Shapiro and Wilk (1965) normality test, with a confidence interval of 95%, confirmed the null hypothesis that the results compared, ILS ($W = 0.75781,\ p = 0.001572$) and the method of Haddadi et al. (2015) ($W = 0.75315,\ p = 0.001391$), could be modeled according to a normal distribution. The *Student's t-test* (Student, 1908) was applied and indicated that there is a significant difference and that ILS has the best mean values ($t = -3.0138,\ df = 13,\ p = 0.004986$) for a significance level of 0.05.

Although the heuristic of Haddadi et al. (2015) addresses the permutation characteristic of CBM, the implementation adopted inevitably converges to a local optimum that is not necessarily a global optimum. Upon coordinating efficient local search procedures with an effective strategy to escape from local optima, the proposed ILS method performs a better sampling of the solution space and outperforms the state-of-the-art method for solving the CBM.

One more time, a version of ILS using random initial solutions was evaluated to assess the ILS convergence. The average gap of random solutions, regarding the solutions of Haddadi et al. (2015) was 33.11%, ranging from 1.45% to 57.48%. On average, the ILS_r required 9.43 iterations to improve the initial solutions in 31.19%. As previously reported, despite the inferior quality of the random initial solutions, ILS_r has converged to solutions with the same quality of its original version without an increase in the number of iterations required. The Shapiro and Wilk (1965) normality test, with a confidence interval of 95%, confirmed the null hypothesis that the results compared, ILS ($W = 0.79402,\ p = 0,0000176$) and ILS_r ($W = 0.79423,\ p = 0,0000178$), could be modeled according to a normal distribution. The *Student's t-test* (Student, 1908) was applied and indicated that there is no a significant difference between the results compared ($t = 0.33277,\ df = 44,\ p = 0.6296$) for a significance level of 0.05.

Time-to-target plots (Aiex et al., 2007), or ttt-plots, were used to illustrate the ILS convergence. The hypothesis behind ttt-plots is

Table 4
Results for artificial instances.

Group	Haddadi et al. (2015)	ILS						
	BKS	*I*	*S**	gap* (%)	*S*	gap (%)	σ	*T*
A	253.00	296.80	**221.40**	−12.49	222.93	−11.89	0.88	100.00
B	695.80	814.40	**634.20**	−8.85	636.74	−8.49	1.15	100.00
C	1358.60	1633.00	**1274.60**	−6.18	1278.02	−5.93	1.73	100.00
D	552.00	632.60	**475.40**	−13.88	478.45	−13.32	1.57	250.06
E	1616.00	1962.40	**1434.80**	−11.21	1442.39	−10.74	3.66	250.06
F	3308.40	4110.80	**3044.60**	-7.97	3057.30	−7.59	6.06	250.01
G	1072.40	1219.80	**902.20**	−15.87	908.84	−15.25	3.86	500.41
H	3125.40	3901.60	**2747.00**	−12.11	2757.92	−11.76	4.99	500.41
I	6375.40	8110.40	**5889.20**	−7.63	5904.16	−7.39	8.16	500.40

Table 5
Results for real-world instances.

Instance	Haddadi et al. (2015)	ILS						
	BKS	*I*	*S**	gap* (%)	*S*	gap (%)	σ	*T*
$RCOV_{1KM}$	**757.00**	757.00	**757.00**	0.00	**757.00**	0.00	0.00	281.64
$RCOV_{2KM}$	**1206.00**	1210.00	1210.00	0.33	1210.00	0.33	0.00	444.30
$RCOV_{3KM}$	1461.00	1449.00	**1449.00**	−0.82	**1449.00**	−0.82	0.00	444.29
$RCOV_{5KM}$	1143.00	1179.00	**1132.00**	−0.96	1139.45	−0.31	5.63	296.10
$RCOV_{10KM}$	280.00	287.00	**276.00**	−1.43	**276.00**	−1.43	0.00	82.00

Table 6
Comparative computational results.

Instance	Concorde	BKS	gap (%)	Time (s)
A	217.8	221.4[b]	1.65	22.51
B	626.2	634.2[b]	1.28	4.81
C	1259.6	1274.6[b]	1.19	43.66
D	472.8	475.4[b]	0.55	3.22
E	1403.8	1434.8[b]	2.21	5.23
F	2997.2	3045.6[b]	1.61	5.14
G	876.8	902.2[b]	2.90	10.23
H	2637.8	2747.0[b]	4.14	5765.71
I	5696.8	5889.2[b]	3.38	5768.33
$RCOV_{1km}$	757	757[a]	0.00	13.48
$RCOV_{2km}$	1196	1206[a]	0.84	14.22
$RCOV_{3km}$	1420	1449[b]	2.04	12.03
$RCOV_{5km}$	1129	1132[b]	0.27	6.89
$RCOV_{10km}$	276	276[b]	0.00	1.52

[a] Haddadi et al. (2015).
[b] ILS.

that a method's running time adjusts to an exponential distribution if the method is executed enough times. Six instances were randomly selected to generate the ttt-plots. For each instance, ILS was run independently 100 times and reported the time needed to reach a target solution value of no more than 5% greater than the best-known solution for the instance. The resulting distribution (*empirical*, represented by the crosses) was compared with the exponential distribution (*theoretical*, represented by the line). The ttt-plots show the cumulative probability (*y* axis) of the method reach a target solution over time(*x* axis).

Fig. 3 (a) - (f) illustrates the convergence of ILS for six instances. All graphs are interpreted in an analogous way. For example, for the artificial instance A_4, illustrated in (a), it is possible to observe that the probability of ILS finding a solution as good as the target value in 7 seconds is 99%, confirming the efficiency of the second established stopping criterion, the dynamic limit on running time.

4.3. Comparison with the exact method

Table 6 presents the solutions found using the Concorde solver, the best known solution value (*BKS*), including the solution values found by the ILS, the percentage distance (gap (%)) between the solutions calculated as $100 \times \frac{\text{BKS - Concorde}}{\text{Concorde}}$, and the Concorde running time. The values for groups *A* through *I* are average values of the five instances solved in each group. Individual results can be found in the supplementary material.

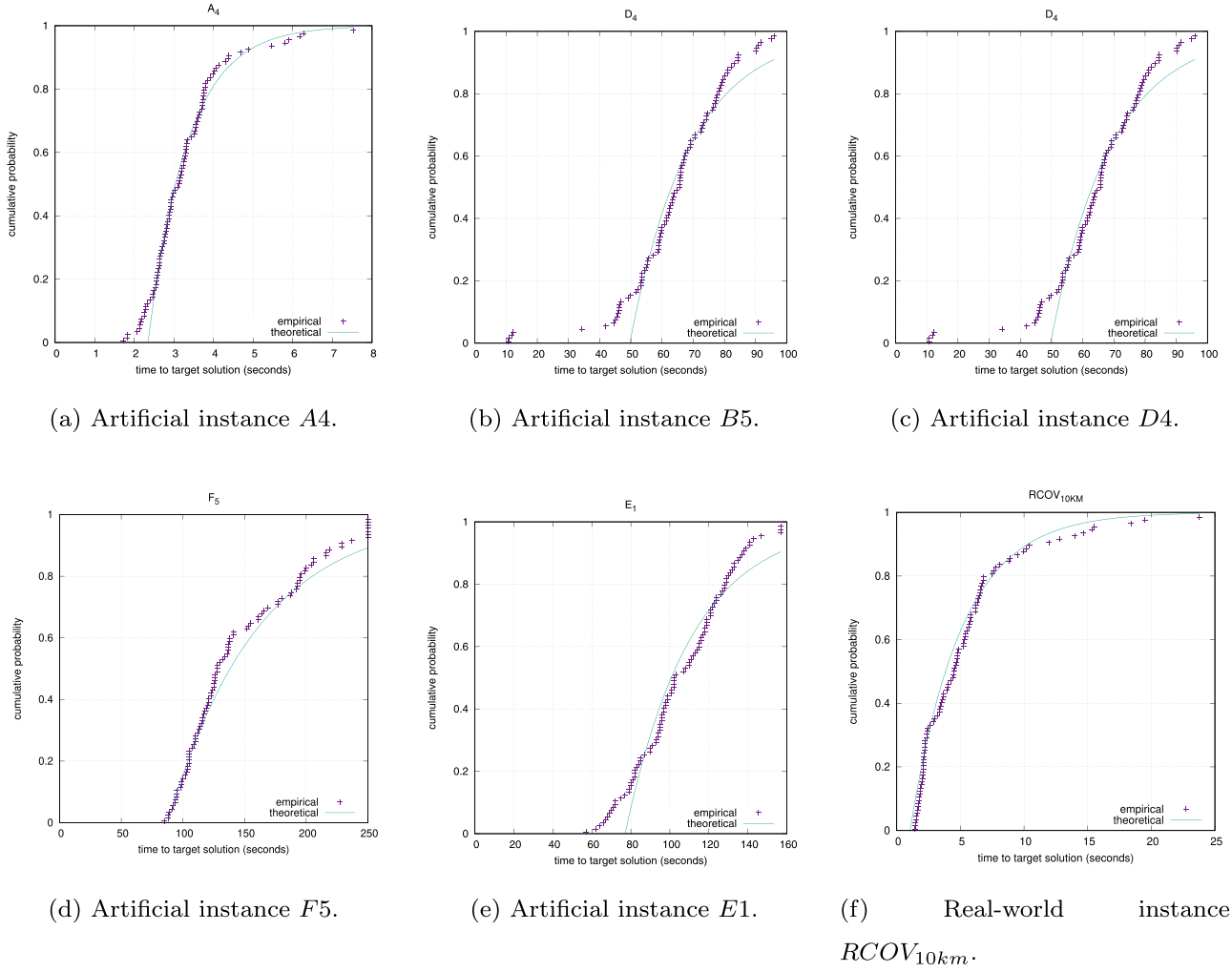
Optimal solutions were found for every instance except four instances: two from group H and two from group I. In these cases, the running time limit of 4 h was reached. The reported results, however, improved upon the best-known solutions. Only two solutions, both from real-world instances, had optimal solutions found prior to this work; however, only one of them ($RCOV_{1km}$) was proven optimal.

The best-known solutions for all instances are, on average, 2.08% distant from the solutions found by the ILS, ranging from 0.00% to 13.50%. The gap of 13.50% was reported for only one instance belonging to the G group. If we leave out this instance, the maximum gap reported is of 3.74%. These low gaps reported evidence the high quality of the solutions achieved by the ILS.

The running time of CBM-TSP reduction is negligible. Each instance is transformed in 0.3 s on an average. We may add to this the running time of the Concorde solver, which is lower than 10 s for most instances. Only seven instances required more computational effort: three instances were solved within 2 min and the other four reached the time limit, which spiked the average running times of groups H and I. A correlation of running time and matrix dimensions or density was not possible because the running times were vastly different among the instances, even those with the same combination of dimensions and density.

5. Conclusions and future work

The CBM is an $\mathcal{NP}$-hard problem with applications in a variety of contexts. In this article, we have proposed a new graph representation for the CBM, a fast constructive heuristic method, the application of the ILS metaheuristic to solving the CBM and, for the first time, we analyze a known method of reducing the CBM problem to the TSP and implement it to find optimal solutions. The quality of the proposed methods has been assessed considering the only benchmark set of instances available in the literature and the current state-of-art method for solving the CBM. The computational experiments performed have shown that the new graph representation and the constructive heuristic method were effective at generating good initial solutions and the ILS consistently



(a) Artificial instance $A4$. (b) Artificial instance $B5$. (c) Artificial instance $D4$.

(d) Artificial instance $F5$. (e) Artificial instance $E1$. (f) Real-world instance $RCOV_{10km}$.

Fig. 3. Illustrative ttt-plots.

generated high-quality solutions in a reasonable running time. Except for four instances, the proposed exact method found optimal solutions. Considering the results and the statistical analysis performed, the proposed methods outperforms the current state-of-art approach for solving the CBM. Suggestions for future research regard important topics such as the development of procedures to obtain good lower bounds for the cases where optimal solutions are not practical and generating new sets of chalenging instances of larger dimensions and different structures.

CRediT authorship contribution statement

Leonardo C.R. Soares: Data curation, Formal analysis, Investigation, Software, Validation, Visualization, Writing - original draft, Writing - review & editing. **Jordi Alves Reinsma:** Formal analysis, Investigation, Software, Validation, Visualization, Writing - review & editing. **Luis H.L. Nascimento:** Investigation, Software, Visualization, Writing - original draft. **Marco A.M. Carvalho:** Conceptualization, Data curation, Investigation, Methodology, Project administration, Supervision, Writing - original draft, Writing - review & editing.

Acknowledgments

Funding: This work was supported by the National Council of Technological and Scientific Development (Conselho Nacional de Desenvolvimento Científico e Tecnológico, CNPq); and Universidade Federal de Ouro Preto. The authors wish to convey their appreciation to Salim Haddadi, who kindly provided the instances used in the computational experiments. This article is dedicated to the loving memory of Celso Francisco Soares.

Supplementary material

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.cor.2020.104948.

References

Aiex, R.M., Resende, M.G., Ribeiro, C.C., 2007. Ttt plots: a perl program to create time-to-target plots. Optim. Lett. 1 (4), 355–366. (2007)

Alizadeh, F., Karp, R.M., Newberg, L.A., Weisser, D.K., 1995. Physical mapping of chromosomes: A combinatorial problem in molecular biology. Algorithmica 13 (1), 52–76. (1995)

Applegate, D., Bixby, R., Chvatal, V., Cook, W., (2006). Concorde TSP solver. http://www.math.uwaterloo.ca/tsp/concorde/.

Atkins, J.E., Boman, E.G., Hendrickson, B., 1998. A spectral algorithm for seriation and the consecutive ones problem. SIAM Journal on Computing 28 (1), 297–310. (1998)

Atkins, J.E., Middendorf, M., 1996. On physical mapping and the consecutive ones property for sparse matrices. Discrete Applied Mathematics 71 (1-3), 23–40. (1996)

Becceneri, J.C., Yanasse, H.H., Soma, N.Y., 2004. A method for solving the minimization of the maximum number of open stacks problem within a cutting process. Comput. Oper. Res. 31 (14), 2315–2332. (2004)

Carvalho, M.A.M., Soma, N.Y., 2015. A breadth-first search applied to the minimization of the open stacks. J. Oper. Res. Soc. 66 (6), 936–946. (2015)

Chakhlevitch, K., Glass, C.A., Shakhlevich, N.V., 2013. Minimising the number of gap-zeros in binary matrices. European Journal of Operational Research 229 (1), 48–58. (2013)
Christof, T., Oswald, M., Reinelt, G., 1998. Consecutive ones and a betweenness problem in computational biology. In: International Conference on Integer Programming and Combinatorial Optimization. Springer, pp. 213–228. (1998)
Christofides, N., 1976. Worst-case analysis of a new heuristic for the travelling salesman problem. Tech. rep., Carnegie-Mellon Univ Pittsburgh Pa Management Sciences Research Group
Crama, Y., Kolen, A.W.J., Oerlemans, A.G., Spieksma, F.C.R., 1994. Minimizing the number of tool switches on a flexible machine. International Journal of Flexible Manufacturing Systems 6 (1), 33–54. (1994)
Croes, G.A., 1958. A method for solving traveling-salesman problems. Operations research 6 (6), 791–812. (1958)
Dom, M., 2009. Recognition, generation, and application of binary matrices with the consecutive ones property. Cuvillier Gottingen. (2009)
Dyson, R., Gregory, A., 1974. The cutting stock problem in the flat glass industry. J. Oper. Res. Soc. 25 (1), 41–53. (1974)
Ernvall, J., Katajainen, J., Penttonen, M., 1985. NP-completeness of the hamming salesman problem. BIT Numer. Math. 25 (1), 289–292. (1985)
Fulkerson, D., Gross, O., 1965. Incidence matrices and interval graphs. Pacific J. Math. 15 (3), 835–855. (1965)
Garey, M.R., Johnson, D.S., 1979. Computers and Intractability:. A guide to the theory of np-completeness.
Haddadi, S., 2002. A note on the np–hardness of the consecutive block minimization problem. Int. Trans. Oper. Res. 9 (6), 775–777. (2002)
Haddadi, S., Chenche, S., Cheraitia, M., Guessoum, F., 2015. Polynomial-time local-improvement algorithm for consecutive block minimization. Information Processing Letters 115 (6), 612–617. (2015)
Haddadi, S., Layouni, Z., 2008. Consecutive block minimization is 1.5-approximable. Information Processing Letters 108 (3), 132–135. (2008)
Hsu, W.-L., Connell, R.M.M., 2003. Pc trees and circular-ones arrangements. Theoretical computer science 296 (1), 99–116. (2003)
Johnson, D., Krishnan, S., Chhugani, J., Kumar, S., Venkatasubramanian, S., 2004. Compressing large boolean matrices using reordering techniques. In: Proceedings of the Thirtieth international conference on Very large data bases-Volume 30, VLDB Endowment, pp. 13–23. (2004)
Kendall, D., 1969. Incidence matrices, interval graphs and seriation in archeology. Pacific J. Math. 28 (3), 565–570. (1969)
Kou, L.T., 1977. Polynomial complete consecutive information retrieval problems. SIAM Journal on Computing 6 (1), 67–75. (1977)
Lima, J.R., Carvalho, M.A.M., 2017. Descent search approaches applied to the minimization of open stacks. Comput. Ind. Eng. 112, 175–186. (2017)
Linhares, A., Yanasse, H.H., 2002. Connections between cutting-pattern sequencing. VLSI Des. Flexib. Mach. Comput. Oper. Res. 29 (12), 1759–1772. (2002)
López-Ibáñez, M., Dubois-Lacoste, J., Cáceres, L.P., Birattari, M., Stützle, T., 2016. The irace package: Iterated racing for automatic algorithm configuration. Operations Research Perspectives 3, 43–58. (2016)
Lourenço, H.R., Martin, O.C., Stützle, T., 2019. Iterated local search: framework and applications. In: Handbook of metaheuristics. Springer, pp. 129–168. (2019)
Madsen, O.B., 1979. Glass cutting in a small firm. Mathematical Programming 17 (1), 85–90. (1979)
Madsen, O.B., 1988. An application of travelling-salesman routines to solve pattern-allocation problems in the glass industry. J. Oper. Res. Soc. 39 (3), 249–256. (1988)
Mittelmann, H., (2016). NEOS server: concorde. https://neos-server.org/neos/solvers/co:concorde/TSP.html.
Paiva, G.S., Carvalho, M.A.M., 2017. Improved heuristic algorithms for the job sequencing and tool switching problem. Comput. Oper. Res. 88, 208–219. (2017)
Reinelt, G., 1994. The Traveling Salesman: computational solutions for TSP applications. Springer-Verlag. (1994)
Shapiro, S.S., Wilk, M.B., 1965. An analysis of variance test for normality (complete samples). Biometrika 52 (3/4), 591–611. (1965)
Student, 1908. The probable error of a mean. Biometrika 6 (1), 1–25. (03 1908)