Modeling Biproportional Apportionment via zero-one matrices with given line sums

Isabella Lari[1], Federica Ricca[1], Andrea Scozzari[2]

[1] Sapienza, Università di Roma,
{isabella.lari, federica.ricca}@uniroma1.it

[2] Università degli Studi "Niccolò Cusano", Roma
andrea.scozzari@unisu.it

July 25, 2012

Abstract

In the Biproportional Apportionment Problem (BAP) a matrix of the vote counts of the parties within the constituencies is given, and one has to convert the vote matrix into an integer matrix of seats "as proportional as possible" to it, subject to the following constraints: i) each constituency must be granted its pre-specified number of seats; ii) each party must be allotted the total number of seats it is entitled to on the basis of its national vote count; iii) a zero-vote zero-seat condition must be satisfied. The matrix of seats must simultaneously meet the integrality and the proportionality requirements and this not infrequently gives rise to self-contradictory procedures in the electoral laws of some countries, such as, for example, Italy. Correct procedures exist for BAP but they are generally not suitable to be written in an electoral law and simpler procedures are needed in order to be acknowledged by legislators and citizens. Without the zero-vote zero-seat condition, that is, when all parties receive some votes in each constituency, BAP can be solved via a very simple procedure based on a central result of the theory of $(0,1)$-matrices with given line sums provided by Gale and Ryser in the 60's. In fact, once the floor of the quota has been assigned to each party in each constituency, basing on the remainders, a very simple rule can be applied to complete the seat matrix with the residual seats. On the contrary, the presence of zeros in the input vote matrix forces zero cells also in the output matrix (fixed zeros) and the previous procedure does not apply any more. In this paper we consider the problem under the fixed zeros condition and analyze specific configurations of the vote matrix for which we show that a modified version of the Ryser procedure works well. For these cases we also state necessary and sufficient conditions for the existence of a feasible solution. Although our analysis does not cover all possible cases, we provide new results on $(0,1)$-matrices with given line sums and fixed zeros which has a central role in the more general area of combinatorial matrix theory.

Keywords: $(0,1)$-matrices with given line sums, matrices with fixed zeros, biproportional apportionment.

1 Introduction

Let m and n be positive integers, $M = \{1,\ldots,m\}$, $N = \{1,\ldots,n\}$, and $r = (r_1, r_2, \ldots, r_m)$ and $s = (s_1, s_2, \ldots, s_n)$ be nonnegative integral vectors satisfying $\sum_{i=1}^{m} r_i = \sum_{j=1}^{n} s_j$. Consider the class of $(0,1)$-matrices with row sum vector r and column sum vector s, and denote it by

$\mathcal{A}(r,s)$. Let Z denote a set of forbidden positions in an $m \times n$ matrix where 1's are not permitted. We are interested in the following *Matrix Feasibility Problem* (MFP): find - if exists - a matrix $A = [a_{ij}]$ of size m by n with row sum vector r and column sum vector s such that $a_{ij} = 0$ for all $(i,j) \in Z$.

The problem was first studied independently in [12] and in [16] for the class $\mathcal{A}(r,s)$ without forbidden cells (i.e., with $Z = \emptyset$). For this class, feasibility conditions were provided and the well-known Ryser's algorithm was introduced to efficiently find a matrix in $\mathcal{A}(r,s)$ (see, for example, [17]). The results by Ryser were further studied and extended by Fulkerson in [11] where the possibility of forbidden cells (or, *fixed zeros*) in $\mathcal{A}(r,s)$ is considered for the first time in the special case of $n \times n$ $(0,1)$-matrices with zero trace. Due to the very particular and regular structure of such matrices, in his paper Fulkerson shows that existence conditions analogous to those provided by Ryser can be stated for matrices constrained to have zero trace, but he does not provide any algorithm for finding one of them. These results were further generalized by Anstee [2] and Chen [9] who extended the analysis to the case of matrices with at most one 0 in each column. In [6] Brualdi et al. study $(0,1)$-matrices of size $m \times n$ with given line sums and a block of zeros (of fixed size) located at one of the corners of the matrix. The existence of such a matrix is characterized in terms of the "structure matrix" already defined in [16] and a modified version of the well-known algorithms by Gale [12] and Ryser [16] is provided. In [7] additional results are provided for matrices showing a configuration of forbidden cells that forms a *Young diagram*. All the above results are collected and further developed in the book published by Brualdi in 2006 which provides a comprehensive review on problems related to combinatorial matrix classes, including MFP [5].

The Matrix Feasibility Problem stated above arises in different applications, such as in graph theory [11] or in the area of discrete tomography [6, 7]. Our work was motivated by an application to the *Biproportional Apportionment Problem* (BAP) which consists of allocating seats to parties in different electoral constituencies having a fixed number of seats at stake in each of them and a fixed number of total (national) seats to assign to each party (see, for example [18]). Given an $m \times n$ matrix V of votes for the parties (n columns) within the constituencies (m rows), one must compute the number of seats that each party should receive in each constituency on the basis of the number of votes that it has obtained in that constituency. The solution of BAP is an integer matrix satisfying both row and column totals and as proportional as possible to the vote matrix V (apportionment) (see, [3, 4]). In many electoral laws, as in the Italian Law for the election of the Chamber of Deputies, the procedure for assigning seats is based on the proportional method by Hare [13]. In order to pursue proportionality, the exact (fractional) quota q_{ij} of seats that a party j should receive in constituency i is first computed (the corresponding matrix $Q = [q_{ij}]$ represents the ideal - but not integral - allocation of seats, see [13, 18]), and a number of seats equal to $\lfloor q_{ij} \rfloor$ is assigned to it in that constituency. The rest of the seats, that necessarily remain in each constituency (*residual seats*), are then assigned to the parties in each constituency according to the non increasing order of their remainders $< q_{ij} >= q_{ij} - \lfloor q_{ij} \rfloor$. It must be noticed that this computation does not guarantee that the output matrix satisfies column (party) sums as required by BAP. Therefore, additional operations are performed in order to meet these conditions. Unfortunately, due to the difficulties of lawmakers to understand the details of the mathematical problem behind BAP, it frequently happens that electoral procedures for this problem are self-contradictory. Correct procedures exist but they are generally not suitable to be written in an electoral law, while simple procedures for BAP are needed that could be acknowledged by legislators and citizens. The results presented in this paper were motivated by the necessity of studying a new BAP procedure for the Italian electoral law for the Chamber of Deputies being both correct and simple at the same time.

Reasoning out about possible simple procedures, we found that the theory of $(0,1)$-matrices with fixed row and column sums may fit our case. Without modifying the first stage of the seat allocation rule described above, one may suggest assigning the floor of the quota to each party in each constituency in order to meet the principle of proportionality between the vote and the seat matrix. After this, one may solve BAP by searching for a *rounding* of Q according to the definition provided by Cox and Ernst in [8]. This means that each party will receive in each constituency at most one additional seat on the basis of his remainder [1]. In view of this, BAP reduces to filling in an $m \times n$ $(0,1)$-matrix $A = [a_{ij}]$ under the constraints that row and column sums must meet the total number of residual seats, in each constituency and for each party, respectively. The final number of seats for party j in constituency i will then be given by $\lfloor q_{ij} \rfloor + a_{ij}$.

In [18] it is illustrated how, under the Cox and Ernst Controlled Rounding model, BAP constraints can be reformulated as those of a capacitated linear transportation problem with forbidden routes. Consider the nonnegative $m \times n$ real matrix Q of regional quotas, and suppose to search for an apportionment that is a rounding of Q and minimizes for example the L_1-distance from Q. The problem can be modeled via the introduction of binary variables a_{ij}, as follows:

$$\begin{array}{lll}
\min \sum_{i\in M} \sum_{j\in N} | \lfloor q_{ij} \rfloor + a_{ij} - q_{ij} | & & \\
\sum_{j\in N} (\lfloor q_{ij} \rfloor + a_{ij}) = r_i & i \in M & \\
\sum_{i\in M} (\lfloor q_{ij} \rfloor + a_{ij}) = s_j & j \in N & \\
a_{ij} = 0 & (i,j) \in \bar{Z} & \\
a_{ij} \in \{0,1\} & i \in M, j \in N, (i,j) \notin \bar{Z} &
\end{array} \tag{1}$$

where M and N denote the set of electoral constituencies (rows) and parties (columns), respectively, while the set $\bar{Z}$ is the set Z (entries with zero votes) plus the entries (if any) with integral q_{ij}.

Following [18], problem (1) can be equivalently written as

$$\begin{array}{lll}
\min \sum_{i\in M} \sum_{j\in N} d_{ij}\, a_{ij} + \sum_{i\in M} \sum_{j\in N} <q_{ij}> & & \\
\sum_{j\in N} a_{ij} = \bar{r}_i & i \in M & \\
\sum_{i\in M} a_{ij} = \bar{s}_j & j \in N & \\
a_{ij} = 0 & (i,j) \in \bar{Z} & \\
0 \le a_{ij} \le 1 & i \in M, j \in N, (i,j) \notin \bar{Z} &
\end{array} \tag{2}$$

where $<q_{ij}>= q_{ij} - \lfloor q_{ij} \rfloor$, and

[1] Notice that in BAP a rounding corresponds to an apportionment satisfying the Hare property.

$$\begin{aligned} d_{ij} &= 1 - 2 <q_{ij}> \\ \bar{r}_i &= r_i - \sum_{j \in N} \lfloor q_{ij} \rfloor \\ \bar{s}_j &= s_j - \sum_{i \in M} \lfloor q_{ij} \rfloor \end{aligned} .$$

Models (1) and (2) differ just by a constant term in the objective function and by the integrality constraints $a_{ij} \in \{0,1\}$ which in (2) can be relaxed into the bounds $0 \leq a_{ij} \leq 1$ thanks to the property of the coefficient matrix of (2) to be totally unimodular.

Following a similar approach, in this paper we still exploit the idea of assigning first $\lfloor q_{ij} \rfloor$ to party j in constituency i, but we resort to the theory of $(0,1)$-matrices with fixed row and column sums and fixed zeros in order to find a rounding of Q without using network flow techniques.

It is well known that even the existence of a feasible $(0,1)$-matrix with forbidden positions can be decided in polynomial time via the solution of a maximum flow problem on a suitable network [1]. However, the results by Gale and Ryser have shown that, when $Z = \emptyset$, a set of n conditions is sufficient to check whether such matrix exists or not, leading to a $O(mn)$ time algorithm for solving the problem. Following the notation in [5], we formulate our Matrix Feasibility Problem as follows. Let $C = [c_{ij}]$ be an $m \times n$ nonnegative integral $(0,1)$-matrix, and let $r = (r_1, r_2, \ldots, r_m)$ and $s = (s_1, s_2, \ldots, s_n)$ be nonnegative integral vectors [2] satisfying $\sum_{i=1}^m r_i = \sum_{j=1}^n s_j$. The set of feasible solutions to MFP with fixed zeros, which we denote by $\mathcal{A}_C(r,s)$, corresponds to the feasible flows in the above network for which the arc capacities are the elements of the matrix C. Notice that when all cells in C are equal to 1, the problem reduces to the one without fixed zeros introduced by Gale and Ryser. In the more general case, C has a 1 only in the positions out of Z.

In this paper we extend the results in [11] assuming that $m = n = qp$ and that the fixed zeros of the matrix correspond exactly to the p square submatrices of order q on the main diagonal. In the BAP application, such zeros correspond to cells (i,j) were party j cannot gain seats in constituency i since it did not receive any vote there. This is the well known *zero votes-zero seats* condition (see, for example, [18]) and the zeros in the p principal square submatrices of order q mean that specific groups of parties did not present their lists in specific groups of constituencies.

As in [11], we also assume that the row and column sum vectors are non increasing and we refer to this problem as *Matrix Feasibility Problem with zero q-blocks* (q-MFP). For this problem we provide necessary and sufficient existence conditions for a feasible solution and we present a $O(n^2)$ time solution algorithm for the special case $q = 1$ (1-MFP).

The paper is organized as follows. In Section 2, we recall some basic results from the literature on MFP and develop new ones related to the specific case of q-MFP. Basing on these results, in Section 3, we provide necessary and sufficient conditions for the existence of a feasible solution for q-MFP, while in Section 4 we illustrate an algorithm for the particular case of 1-MFP. Finally, in Section 5, we draw some conclusions and discuss possible lines of research.

2 Basic results

In MFP we are given two nonnegative integral vectors $r = (r_1, r_2, \ldots, r_m)$ and $s = (s_1, s_2, \ldots, s_n)$ such that $\sum_{i=1}^m r_i = \sum_{j=1}^n s_j = \tau$ and an $m \times n$ capacity $(0,1)$-matrix $C = [c_{ij}]$. We investigate the existence conditions of an $m \times n$ $(0,1)$-matrix A in $\mathcal{A}_C(r,s)$. Relying on the theory of

[2]In the BAP application r and s denote the row and column sums of the residual seats.

network flows [1], this problem can be formulated and solved as a maximum flow-minimum cut problem on a suitable network G and necessary and sufficient conditions for the existence of a matrix in $\mathcal{A}_C(r, s)$ can be derived. Actually, it can be shown [10] that there is a one-to-one correspondence between cuts in G and pairs (R, S), where R is a subset of the set of rows M and S is a subset of the set of columns N and that the capacity $C(R, S)$ of the cut corresponding to (R, S) is given by

$$C(R,S) = 2\tau - \sum_{i\in R} r_i - \sum_{j\in S} s_j + \sum_{i\in R}\sum_{j\in S} c_{ij}.$$

A necessary and sufficient condition for the existence of a matrix in $\mathcal{A}_C(r, s)$ is that the minimum capacity of a cut in G is equal to τ, that is,

$$C(R,S) \geq \tau, \quad \forall\, R, S \tag{3}$$

or, equivalently,

$$\tau - \sum_{i\in R} r_i - \sum_{j\in S} s_j + \sum_{i\in R}\sum_{j\in S} c_{ij} \geq 0, \quad \forall\, R, S. \tag{4}$$

The above exponential number of conditions were already pointed out independently by Gale [12] and Ryser [16] in 1957, and, for the case $c_{ij} = 1, i \in M, j \in N$, they both showed that a linear number of conditions is enough. These conditions require to check that the column sum vector s is majorized (in the sense of Hardy-Littlewood-Pólya [14]) by the vector of the conjugate sequence of the row sum vector r (see, [16] Chapter 6, Theorem 1.1).

For the sake of simplicity we denote

$$g(R,S) = \sum_{i\in R} r_i + \sum_{j\in S} s_j - \sum_{i\in R}\sum_{j\in S} c_{ij} \geq 0, \quad \forall\, R, S. \tag{5}$$

To check feasibility for MFP, one can either minimize the left-hand side of (4) or maximize $g(R, S)$ over all pairs R, S, verifying that for an optimal solution R^*, S^*, one has $g(R^*, S^*) = \tau$.

On the basis of the above results, both Gale and Ryser provided polynomial time algorithms for MFP without fixed zeros, which are conceptually simpler than network flow techniques. Fulkerson [11], Brualdi and Dahl [6, 7], and others, already provided some results on MFP for special cases in which the matrix C has a particular configuration but the general problem still lacks of a unifying existence result.

In his paper Fulkerson considered the case of square matrices having zero trace and non increasing row and column sums. In the present paper we start from the results by Fulkerson and generalize them to the case $m = n = qp$ and the fixed zeros of the matrix are in the p principal $q \times q$ submatrices located in the main diagonal of C. This case is not still covered by previous results in the literature. The following theorem is provided in [5] (see, Theorem 4.4.2 in Section 4) and seems to be the most general result for MFP with fixed zeros. Note that the original version of the theorem is more general than our $(0, 1)$-matrix case, since it refers to the existence of integral matrices with fixed zeros.

Theorem 2.1 *(See, [5]) Let $r = (r_1, r_2, \ldots, r_m)$ and $s = (s_1, s_2, \ldots, s_n)$ be nonnegative integral vectors satisfying*

$$\sum_{i=1}^{m} r_i = \sum_{j=1}^{n} s_j. \tag{6}$$

Let $C = [c_{ij}]$ be a nonnegative integral matrix of size m by n satisfying

$$\sum_{i=1}^{m} (c_{ij} - c_{ik})^+ \leq s_j - s_k + 1 \qquad 1 \leq j < k \leq n. \tag{7}$$

There exists a matrix $A = [a_{ij}]$ in $\mathcal{A}_C(r, s)$ if and only if

$$\sum_{i=1}^{m} (r_i - \sum_{j=1}^{h} c_{ij})^+ \leq \sum_{j=h+1}^{n} s_j \qquad h = 1, \ldots, n. \tag{8}$$

The above theorem states that (8) is a necessary and sufficient condition for the existence of A in $\mathcal{A}_C(r, s)$, provided that C satisfies condition (7). As noted by Brualdi, in the theorem it is not assumed that s is non increasing, but condition (7) implies a sort of *nearly* non increasing property of S, in the sense that, under condition (7) for $j < k$ one has $s_j \geq s_k - 1$ (see, [5]).

It is easy to check that the case of at most one zero in each column of C is solved by this theorem, while the following example shows that in our version of MFP, characterized by $q \times q$ "blocks" of fixed zeros, the above theorem does not apply any more, and in this case a specific result is required to provide necessary and sufficient conditions for the existence of a matrix A in $\mathcal{A}_C(r, s)$.

Example 1

Consider the following 6×6 matrix C, that in every matrix in $\mathcal{A}_C(r, s)$ forces to zero all cells in the two diagonal square matrices of order $q = 3$, and assume $r_i = 1$ for all $i = 1, \ldots, m$, $s_j = 1$ for all $j = 1, \ldots, n$:

$$C = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \end{bmatrix}.$$

The instance of MFP is feasible and a feasible solution in $\mathcal{A}_C(r, s)$ is given by

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

but matrix C does not satisfy condition (7). In fact, for a pair of columns j and k, $1 \leq j \leq 3$ and $4 \leq k \leq 6$, one has:

$$\sum_{i=1}^{m} (c_{ij} - c_{ik})^+ = 3 > 1 = s_j - s_k + 1.$$

We now focus on q-MFP. For $h = 1, \dots, p$, we define

$$R_h = \{(h-1)q + 1, \dots, qh\}$$

and

$$S_h = \{(h-1)q + 1, \dots, qh\}$$

corresponding to the row and column indexes of the h-th $q \times q$ submatrix in the main diagonal of A.
In the following a set of rows R_h will be called a *row block* and a set of columns S_h a *column block*, while the pair (R_h, S_h) will be simply referred to as a *block*.

According to our specific formulation of q-MFP, we assume that

$$\begin{cases} c_{ij} = 0 & \text{for all pairs } (i,j) \text{ such that } i \in R_h \text{ and } j \in S_h, h = 1, \dots, p \\ c_{ij} = 1 & \text{otherwise} \end{cases} \tag{9}$$

and

$$\begin{cases} r_i \geq r_{i+1} & i = 1, \dots, n-1 \\ s_j \geq s_{j+1} & j = 1, \dots, n-1. \end{cases} \tag{10}$$

The main results of this paper rely on assumptions (9) and (10). However, for some minor results (10) can be replaced by the following weaker monotonicity conditions:

$$\begin{cases} r_i \geq r_{i+1} & i \in R_h \backslash \{qh\}, h = 1, \dots, p \\ s_j \geq s_{j+1} & j \in S_h \backslash \{qh\}, h = 1, \dots, p \end{cases} \tag{11}$$

The following Proposition 2.2 states a nice property of the sets R and S that maximize function $g(R, S)$ introduced in (5).

Proposition 2.2 *Suppose that hypothesis (9) and (11) hold. Then, among all pairs of row and column subsets (R, S) with $|R_h \cap R| = k_h \geq 1$ and $|S_h \cap S| = l_h \geq 1$, $h = 1, \dots, p$, there exists one, say, (R^*,S^*), maximizing function $g(\cdot)$ and such that:*

i) for each row block R_h including at least one row in R^*, one has:
$R^* \cap R_h = \{(h-1)q + 1, \dots, (h-1)q + k_h\}$
(i.e., R^* contains the first k_h rows of R_h);

ii) for each column block S_h including at least one column in S^*, one has:
$S^* \cap S_h = \{(h-1)q + 1, \dots, (h-1)q + l_h\}$
(i.e., S^* contains the first l_h rows of S_h).

Proof Given a set of rows R and a set of columns S, suppose that $|R_h \cap R| = k_h \geq 1$. If $k_h = q$ the row block R_h satisfies property i), since all rows of R_h belong to R. Suppose now that $k_h \leq q-1$ and R does not satisfy property i). Hence, there exists in R a row $(h-1)q + t$ such that $k_h < t \leq q$, and there exists a row $(h-1)q + z$ which does not belong to R such that $1 \leq z \leq k_h$. Let $R' = R \backslash \{(h-1)q + t\} \cup \{(h-1)q + z\}$. Since rows $(h-1)q + t$ and $(h-1)q + z$ are in the same block h, we have:

$$g(R', S) = g(R, S) - r_{(h-1)q+t} + r_{(h-1)q+z} + \sum_{j \in S} (c_{(h-1)q+t,j} - c_{(h-1)q+z,j}).$$

By hypothesis (9), $c_{(h-1)q+t,j} = c_{(h-1)q+z,j}$ for all $j \in N$, hence:

$$g(R', S) = g(R, S) - r_{(h-1)q+t} + r_{(h-1)q+z}.$$

Moreover, by hypothesis (11), $r_{(h-1)q+t} \leq r_{(h-1)q+z}$, so that

$$g(R', S) \geq g(R, S).$$

Thus, replacing row $(h-1)q+t$ by row $(h-1)q+z$ in R produces a new pair (R', S) with a better value of $g(\cdot)$ w.r.t. (R, S). We can repeat the above argument in order to finally obtain a set R^* satisfying property $i)$ and having, for any given S, the largest value of $g(\cdot)$ among all pairs (R, S) with $|R_h \cap R| = k_h$. A similar construction can be applied also to the set of columns to finally obtain the pair (R^*, S^*) maximizing $g(\cdot)$ and such that S^* satisfies property $ii)$. □

Replacing (10) by the stronger assumption (9), leads to the following stronger result.

Proposition 2.3 *Suppose that hypothesis (9) and (10) hold. Then, among all pairs of row and column subsets (R, S) with $|R| = k \geq 0$ and $|S| = l \geq 0$, there exists one, say, (R^*,S^*), maximizing function $g(\cdot)$ and such that:*

a) either R^* is empty (if $k = 0$), or $R^* = \{1, \ldots, k\}$, (if $1 \leq k \leq n$);

b) either S^* is empty (if $l = 0$), or $S^* = \{1, \ldots, l\}$, (if $1 \leq l \leq n$).

Proof Suppose we are given the subset of rows R and the subset of columns S, with $|R_h \cap R| = k_h \geq 1$ and $|S_h \cap S| = l_h \geq 1$, $h = 1, \ldots, p$, $\sum_{h=1}^{p} k_h = k$ and $\sum_{h=1}^{p} l_h = l$, maximizing function $g(\cdot)$ and satisfying properties $i)$ and $ii)$ of Proposition 2.2. Such a pair of sets exists for every possible fixed cardinalities k_h and l_h, $h = 1, \ldots, p$, since (10) implies (11). Now suppose that R is not empty and does not satisfy property $a)$. This means that there exist two row blocks $R_{h'}$ and $R_{h''}$ with $h'' > h'$ such that $k_{h'} < q$ and $k_{h''} \geq 1$.

We will show that updating R by including the maximum possible number of rows in block h' in place of the same number of rows in h'' (which will be deleted from R), and performing a similar replacement involving the columns of blocks h' and h'', leads to a new pair (R^1, S^1) such that $g(R^1, S^1) \geq g(R, S)$. The idea of the proof is that, when replacing (R, S) by (R^1, S^1), by assumption (10) the sum of the r_i's and of the s_j's in $g(\cdot)$ increases, while the sum of the c_{ij}'s in $g(\cdot)$ decreases, since, by (9), the number of fixed zeros in this sum increases. By repeatedly applying this updating to the new generated pairs until condition $a)$ holds, and doing the same, in a second phase, for the columns to satisfy property $b)$, one finally gets the optimal pair (R^*, S^*) with the required properties $a)$ and $b)$.

Let $\alpha = min\{q - k_{h'}, k_{h''}\}$ and consider the sets of rows

$$A_{h''} = \{(h''-1)q + k_{h''} - \alpha + 1, \ldots, (h''-1)q + k_{h''}\}$$

and

$$A_{h'} = \{(h'-1)q + k_{h'} + 1, \ldots, (h'-1)q + k_{h'} + \alpha\},$$

for which one has $|A_{h'}| = |A_{h''}| = \alpha$. Let

$$R^1 = R \backslash A_{h''} \cup A_{h'}.$$

Moreover, let $\beta = min\{q - l_{h'}, l_{h''}\}$. If $\beta > 0$ define the sets of columns

$$B_{h''} = \{(h'' - 1)q + l_{h''} - \beta + 1, \dots, (h'' - 1)q + l_{h''}\}$$

and

$$B_{h'} = \{(h' - 1)q + l_{h'} + 1, \dots, (h' - 1)q + l_{h'} + \beta\}$$

with $|B_{h'}| = |B_{h''}| = \beta$, and let

$$S^1 = S \backslash B_{h''} \cup B_{h'}.$$

If $\beta = 0$ let $S^1 = S$ and $B_{h'} = B_{h''} = \emptyset$.

Now we compare $g(R, S)$ and $g(R^1, S^1)$. We have:

$$\begin{aligned} g(R,S) &= \sum_{i \in R} r_i + \sum_{j \in S} s_j - \sum_{i \in R} \sum_{j \in S} c_{ij} \\ &= \sum_{i \in R} r_i + \sum_{j \in S} s_j - |R||S| + \sum_{i \in R} \sum_{j \in S} (1 - c_{ij}) \\ &= \sum_{i \in R} r_i + \sum_{j \in S} s_j - |R||S| + \sum_{h \in \{1,\dots,p\}} k_h l_h \end{aligned}$$

and

$$\begin{aligned} g(R^1,S^1) &= \sum_{i \in R^1} r_i + \sum_{j \in S^1} s_j - \sum_{i \in R^1} \sum_{j \in S^1} c_{ij} \\ &= \sum_{i \in R^1} r_i + \sum_{j \in S^1} s_j - |R||S| + \sum_{i \in R^1} \sum_{j \in S^1} (1 - c_{ij}) \\ &= \sum_{i \in R^1} r_i + \sum_{j \in S^1} s_j - |R||S| + \sum_{h \in \{1,\dots,p\}: h \neq h', h''} k_h l_h \\ &\quad + (k_{h'} + \alpha)(l_{h'} + \beta) + (k_{h''} - \alpha)(l_{h''} - \beta). \end{aligned}$$

Hence,

$$\begin{aligned} g(R^1,S^1) - g(R,S) &= \sum_{i \in A_{h'}} r_i - \sum_{i \in A_{h''}} r_i + \sum_{j \in B_{h'}} s_j - \sum_{j \in B_{h''}} s_j \\ &\quad + \alpha(l_{h'} - l_{h''} + \beta) + \beta(k_{h'} - k_{h''} + \alpha). \end{aligned}$$

Since $|A_{h'}| = |A_{h''}|$, by hypothesis (10) we have $\sum_{i \in A_{h'}} r_i \geq \sum_{i \in A_{h''}} r_i$. Similarly, $\sum_{j \in B_{h'}} s_j \geq \sum_{j \in B_{h''}} s_j$. Moreover, it is easy to verify that $\alpha(l_{h'} - l_{h''} + \beta) \geq 0$ and $\beta(k_{h'} - k_{h''} + \alpha) \geq 0$ for all possible values of α and β. It follows that $g(R^1, S^1) \geq g(R, S)$.

We can repeatedly apply the above argument to row blocks which still do not satisfy property *a*), until, at some step γ, we obtain the set of cardinality k that satisfies property *a*), i.e., $R^\gamma = R^*(k)$. Analogous operations can be applied if the set of columns in the current pair $(R^*(k), S^\gamma(l))$ does not satisfy property *b*). The final pair $(R^*(k), S^*(l))$ satisfies properties *a*) and *b*) and conditions $|R^*(k)| = k$ and $|S^*(l)| = l$. Since the initial pair (R, S) maximizes $g(\cdot)$ under the weaker monotonicity assumption (11) for fixed k_h and l_h, $h = 1, \dots, p$, in view of the fact that $g(R^*(k), S^*(l)) \geq g(R, S)$, it follows that, under the stronger monotonicity assumption (10), $g(R^*(k), S^*(l))$ maximizes $g(\cdot)$ for the corresponding $k = \sum_{h=1}^p k_h$ and $l = \sum_{h=1}^p l_h$. The same procedure leads to an optimal pair $g(R^*(k), S^*(l))$ for every possible fixed k and l. $\square$

3 Existence conditions for a feasible solution of q-MFP

The results of Section 2 can be exploited in different ways to state necessary and sufficient conditions for the existence of a feasible solution for q-MFP.

First, in view of Proposition 2.3, we can define a *structure matrix* $T(r, s, q)$ for the class $\mathcal{A}_C(r, s)$ where C is defined by (9). As in the structure matrix already introduced by Ford and Fulkerson (see, [10]), the generic element t_{kl} of $T(r, s, q)$, $k = 0, 1, \ldots, n$ and $l = 0, 1, \ldots, n$, computes the value $C(R, S) - \tau$, with $R = \{1, \ldots, k\}$ and $S = \{1, \ldots, l\}$. Therefore, conditions (4) can be efficiently checked by verifying $t_{kl} \geq 0$ for $k = 0, 1, \ldots, n$ and $l = 0, 1, \ldots, n$. The generic element of the $(n + 1) \times (n + 1)$ structure matrix $T(r, s, q)$ is given by

$$\begin{aligned} t_{kl} \;&=\; \big[\tau - \sum_{i=1}^{k} r_i\big] - \sum_{j=1}^{l} s_j + kl - \lfloor\tfrac{\min\{k,l\}}{q}\rfloor \cdot q^2 - \big\{\big[\min\{k,l\} - \lfloor\tfrac{\min\{k,l\}}{q}\rfloor \cdot q\big]\times \\ &\quad \times\big[q - \big(q \cdot \big(\lfloor\tfrac{\min\{k,l\}}{q}\rfloor + 1\big) - \max\{k,l\}\big)^+\big]\big\}, \end{aligned} \tag{12}$$

or equivalently:

$$\begin{aligned} t_{kl} \;&=\; \sum_{i=k+1}^{n} r_i - \sum_{j=1}^{l} s_j + kl - \lfloor\tfrac{\min\{k,l\}}{q}\rfloor \cdot q^2 - \big\{\big[\min\{k,l\} - \lfloor\tfrac{\min\{k,l\}}{q}\rfloor \cdot q\big]\times \\ &\quad \times\big[q - \big(q \cdot \big(\lfloor\tfrac{\min\{k,l\}}{q}\rfloor + 1\big) - \max\{k,l\}\big)^+\big]\big\}, \end{aligned} \tag{13}$$

where $a^+ = \max\{a, 0\}$.

It is easy to see that, once $\sum_{i=1}^{k} r_i$ and $\sum_{j=1}^{l} s_j$ have been computed for all k and l, t_{kl} can be obtained in constant time. Hence, one can determine in $O(n^2)$ time whether the class $\mathcal{A}_C(r, s)$ is empty or not. We also point out that in the particular cases $q = 0$ or $q = 1$, the above formulas for t_{kl} have the same expressions already provided by Ford and Fulkerson [10] and Fulkerson [11], respectively.

The results of Proposition 2.3 can be exploited directly to reduce the exponential number of conditions (4) to a linear number of conditions for q-MFP. To obtain this, we observe that, for any fixed subset of columns S, one can rewrite condition (4) as follows:

$$\delta(S) = \min_{R\subseteq\{1,\ldots,n\}} \big[C(R, S) - \tau\big] = \sum_{i=1}^{n} \min\{\sum_{j\in S} c_{ij}, r_i\} - \sum_{j\in S} s_j \geq 0 \quad \forall S \subseteq \{1, \ldots, n\}. \tag{14}$$

We can replace the above set of conditions by a smaller set, since, for all subsets of columns S with the same cardinality l, from Proposition 2.3 we know that $\delta(S) \geq \delta(\{1, \ldots, l\})$, and, therefore, only the set $S = \{1, \ldots, l\}$ of the first l columns is relevant for finding R^* and S^*. Hence, we can restrict to verify only the following n conditions:

$$\delta(\{1, \ldots, l\}) = \sum_{i=1}^{n} \min\{\sum_{j=1}^{l} c_{ij}, r_i\} - \sum_{j=1}^{l} s_j \geq 0 \quad l = 1, \ldots, n. \tag{15}$$

We observe that $\sum_{i=1}^{n} \min\{\sum_{j=1}^{l} c_{ij}, r_i\}$ is the maximum number of 1's that, according to r and C, can be put in the first j columns of the output $(0,1)$-matrix.

Following an approach similar to the one proposed in [16], the above conditions can be rewritten in terms of the conjugate sequence of vector r. In our case, the constraints of the fixed zeros in some positions of the $(0,1)$-matrix require the definition of a *C-conjugate* sequence of r, $r^*_j(C)$, $j = 1,\ldots,n$, that is, the conjugate sequence of r that also takes into account the capacities equal to 0 in the matrix C:

$$r^*_j(C) = |i \in \{1,\ldots,n\} : c_{ij} = 1, r_i \geq \sum_{t=1}^{j} c_{it}|. \tag{16}$$

Notice that, when the capacities are all equal to 1, $r^*_j(C)$, $j = 1,\ldots,n$, corresponds to the standard conjugate sequence of r adopted in [17].

It is easy to see that $\sum_{j=1}^{l} r^*_j(C) = \sum_{i=1}^{n} \min\{\sum_{j=1}^{l} c_{ij}, r_i\}$, so that, using $r^*_j(C)$, conditions (15) become:

$$\sum_{j=1}^{l} r^*_j(C) \geq \sum_{j=1}^{l} s_j \quad l = 1,\ldots,n. \tag{17}$$

For q-MFP, the above conditions can be verified in $O(n^2)$ time, since for every column j one can compute $r^*_j(C)$ in $O(n)$ time following an incremental strategy that allows the updating of $r^*_j(C)$ from $r^*_{j-1}(C)$ in constant time.

4 An algorithm for 1-MFP

In [11] Fulkerson provides necessary and sufficient conditions for the existence of a feasible solution for MFP with fixed zeros on the main diagonal of the output $(0,1)$-matrix which is a special case of q-MFP. However, he did not provide any algorithm for finding a feasible solution for this problem, neither did he suggest any way of exploiting the existence conditions in this sense. It is obvious that a feasible solution can be always found via network flow techniques, but, in this section, we provide an algorithm to practically find a solution of 1-MPF in $O(n^2)$ time, once the existence conditions hold true. The same conditions are used to show the correctness of the algorithm.

One may think that the straightforward application of the classical algorithms by Ryser [16] or by Gale [12] may work also for 1-MPF. The following examples show that this is not true, since although the 1-MPF instances are feasible, the presence of fixed zeros causes the above algorithms to prematurely stop without finding a solution. In Example 2, we apply the algorithm by Ryser. According to it, the "maximal matrix" (see, [16] for details) is computed at the beginning (iteration 0), and then the algorithm moves the 1's from left to right starting from the largest row sums and giving preference to the bottom-most 1 in case of a tie. In the 1-MFP instance of Example 2, the algorithm stops at iteration 3 in column 3 (in bold) due to an excess of 1's in this column w.r.t. the corresponding column sums.

Example 2
Consider the 1-MFP instance with $r = (3,2,2,1,1)$ and $s = (3,3,1,1,1)$ and apply the Ryser procedure [16]. The steps are the following (a "$\times$" denotes a fixed zero):

$$\begin{array}{cccc}
\text{iteration 0} & \text{iteration 1} & \text{iteration 2} & \text{iteration 3} \\
\begin{bmatrix} \times & 1 & 1 & 1 & 0 \\ 1 & \times & 1 & 0 & 0 \\ 1 & 1 & \times & 0 & 0 \\ 1 & 0 & 0 & \times & 0 \\ 1 & 0 & 0 & 0 & \times \end{bmatrix} &
\begin{bmatrix} \times & 1 & 1 & 0 & 1 \\ 1 & \times & 1 & 0 & 0 \\ 1 & 1 & \times & 0 & 0 \\ 1 & 0 & 0 & \times & 0 \\ 1 & 0 & 0 & 0 & \times \end{bmatrix} &
\begin{bmatrix} \times & 1 & 1 & 0 & 1 \\ 1 & \times & 1 & 0 & 0 \\ 1 & 0 & \times & 1 & 0 \\ 1 & 0 & 0 & \times & 0 \\ 1 & 0 & 0 & 0 & \times \end{bmatrix} &
\begin{bmatrix} \times & 1 & \mathbf{1} & 0 & 1 \\ 1 & \times & \mathbf{1} & 0 & 0 \\ 1 & 0 & \times & 1 & 0 \\ 1 & 0 & \mathbf{0} & \times & 0 \\ 1 & 0 & \mathbf{0} & 0 & \times \end{bmatrix}.
\end{array}$$

At iteration 3 the third column has two 1's but its total is $s_3 = 1$ and the algorithm stops without finding a feasible solution. Nevertheless, $\mathcal{A}_C(r,s)$ is non empty and, for example, a feasible solution is the following:

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

The algorithm by Gale starts by filling the matrix from the first column and at iteration j it inserts the 1's in the s_j rows with largest row sums (except for row j in 1-MFP), giving preference to the top-most 1's in case of ties. Example 3 illustrates a case in which the Gale procedure fails in filling the last column (in bold), since, according to it, the last 1 should be placed in the last row which, for 1-MPF, always corresponds to a fixed zero.

Example 3

Consider the 1-MFP instance with $r = (3, 2, 2, 2, 1)$ and $s = (3, 3, 2, 1, 1)$ and apply the Gale procedure [16]. We have:

$$\begin{array}{cccc}
\text{iteration 1} & \text{iteration 2} & \text{iteration 3} & \text{iteration 4} \\
\begin{bmatrix} \times & 0 & 0 & 0 & 0 \\ 1 & \times & 0 & 0 & 0 \\ 1 & 0 & \times & 0 & 0 \\ 1 & 0 & 0 & \times & 0 \\ 0 & 0 & 0 & 0 & \times \end{bmatrix} &
\begin{bmatrix} \times & 1 & 0 & 0 & 0 \\ 1 & \times & 0 & 0 & 0 \\ 1 & 1 & \times & 0 & 0 \\ 1 & 1 & 0 & \times & 0 \\ 0 & 0 & 0 & 0 & \times \end{bmatrix} &
\begin{bmatrix} \times & 1 & 1 & 0 & 0 \\ 1 & \times & 1 & 0 & 0 \\ 1 & 1 & \times & 0 & 0 \\ 1 & 1 & 0 & \times & 0 \\ 0 & 0 & 0 & 0 & \times \end{bmatrix} &
\begin{bmatrix} \times & 1 & 1 & 1 & \mathbf{0} \\ 1 & \times & 1 & 0 & \mathbf{0} \\ 1 & 1 & \times & 0 & \mathbf{0} \\ 1 & 1 & 0 & \times & \mathbf{0} \\ 0 & 0 & 0 & 0 & \times \end{bmatrix}.
\end{array}$$

Even in this case, a feasible solution exists being, for example, the following one:

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

In the following, we present an algorithm for 1-MFP which implements a simple variant of the original strategy of the Ryser algorithm. The idea is always to start with a matrix where in each row i we have 1 in the first r_i positions having capacity equal to 1 and 0 elsewhere and then moving 1's from left to right in those rows having the largest row sums. The difference is that in case of a tie we choose the top-most 1 instead of the bottom-most as in the Ryser algorithm.

Given an n-vector r, we define the $n \times n$ *maximal* matrix $\bar{A}^C$ as the matrix having row sums r_i, $i = 1, \ldots, n$, and column sums $r_j^*(C)$, $j = 1, 2, \ldots, n$. It is easy to see that for each row i of

$\bar{A}^C$ the first r_i elements having capacity different from 0 are equal to 1 and the other elements are equal to 0. For the sake of simplicity, when there is no danger of confusion, we will denote $\bar{A}^C$ simply by $\bar{A}$.

In the TOP-MOST-TIE algorithm, we are going to describe below for 1-MFP, we suppose that condition (17) holds true so that $\mathcal{A}_C(r, s)$ is nonempty. As in the Ryser algorithm, the starting maximal matrix $\bar{A}$ is iteratively modified by moving 1's from left to right in order to satisfy the column sums. At iteration $n-f+1$ the last $n-f$ columns of the matrix are the last $n-f$ columns of the output matrix; while the first f columns define an $n \times f$ maximal matrix. We denote by $\bar{A}_f$ the matrix containing the first f column of $\bar{A}$ and by $\bar{a}_{ij}$ the element in row i and column j of $\bar{A}$. We also denote the column sums of $\bar{A}_f$ by $e_1, \ldots, e_f$ and its row sums by $\bar{r}_1, \ldots, \bar{r}_n$.

ALGORITHM TOP-MOST-TIE
Input: a row sum nonnegative n-vector r and a column sum nonnegative n-vector s satisfying conditions (17). An $n \times n$ capacity matrix C such that $c_{ii} = 0, i = 1, \ldots, n$ and $c_{ij} = 1, i, j = 1, \ldots, n,\ i \neq j$.
Output: A matrix $\bar{A} \in \mathcal{A}_C(r, s)$.

1. build the maximal matrix $\bar{A}$;
2. let $e := r^*(C)$, $\bar{r} := r$, and $f := n$ $(\bar{A}_f = \bar{A})$;
3. update $\bar{A}$ as follows: move to column f the right-most 1's in the s_f rows of $\bar{A}_f$, different from row f, having the largest row sums (note that some of these 1's may already be in column f). In case of a tie between row sums, move first the 1 from the top-most row;
4. update $e_1, \ldots, e_{f-1}$ and $\bar{r}_1, \ldots, \bar{r}_n$ to be the column sums and the row sums of the matrix $\bar{A}_{f-1}$, respectively;
5. let $f := f - 1$;
6. if $f := 1$ STOP and output $\bar{A}$, else go to step 3;

In Theorem 4.3 we will prove that under hypothesis (17) at each iteration of the TOP-MOST-TIE algorithm it is always possible to perform step 3 and fill the current column f with the required number s_f of 1's. We now give some insight on the algorithm behavior that will be useful in the proof of this theorem.

First of all we note that at iteration $n-f+1$ the totals of columns $f+1, \ldots, n$ of $\bar{A}$ are equal to the required values $s_{f+1}, \ldots, s_n$. It follows that the column sums of $\bar{A}_f$, $e_1, \ldots, e_f$, are such that

$$\sum_{j=1}^{f} e_j = \sum_{j=1}^{f} s_j. \tag{18}$$

Also notice that, when $f = 1$ (18) implies $e_1 = s_1$ and the algorithm can stop since the first column already satisfies its total.

Moreover, since the starting matrix is maximal and the 1's that are moved at iteration $n-f+1$ are the right-most 1's in their rows in $\bar{A}_f$, the remaining matrix $\bar{A}_{f-1}$ is a maximal matrix, as well.

To illustrate how the algorithm works, in Figure 1 we show the generic configuration for two consecutive columns k and $k+1$ in $\bar{A}_f$ with $f > k+1$. According to the algorithm strategy,

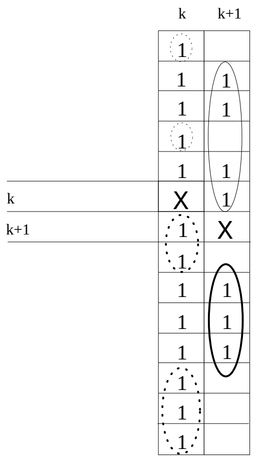

Figure 1: The sequence of the (four) sets of 1's that, according to the TOP-MOST-TIE rule, can be moved from two consecutive columns of $\bar{A}_f$. The first set is in the boldface circle, the second set is in the circle in column $k+1$; the third set is represented by the 1's in the dotted boldface circles of column k, and, finally, the fourth set is given by the 1's in the dotted circles in rows $i = 1, \ldots, k-1$ and column k.

the 1's in column k are moved to the right only if all the 1's in column $k+1$ have been already moved. Indeed, considering the right-most 1 in each row, and choosing the top-most 1 in case of a tie, the first set of 1's to move to column $f > k+1$ is the set of 1's in column $k+1$ and rows $i = k+2, \ldots, n$ (see the 1's in the boldface circle of Figure 1). The second set of 1's to be moved are those in column $k+1$ and in rows $i = 1, \ldots, k$ (the 1's in the circle of Figure 1). Note that the algorithm may be prevented from moving a 1 from cell $(i, k+1)$ to (i, f) whenever $i = f$ or when there is already a 1 in (i, f), but, in any case, if a 1 cannot be moved from column $k+1$ to f, it cannot be moved from column k to f as well. Once all the possible 1's in column $k+1$ have been moved, the next set of 1's to move are those in the boldface dotted circles and then those in the dotted circles (see, Figure 1).

Consider iteration $n-h+1$ of the algorithm, i.e., when column h has to be filled, and let $f < h$. We denote by e_f^{n-h+1} the number of 1's in column f at iteration $n-h+1$.

Lemma 4.1 *If $s_h < e_f^{n-h+1}$ holds, then in order to fill column h and satisfy its total s_h, the algorithm TOP-MOST-TIE needs not to fall upon 1's in columns $g < f$.*

Proof The result follows from two facts: i) the current number of 1's in column f is at least one more than the necessary to fill column h; ii) no 1's can be moved in column h from column $g < f$ before all 1's in f have been moved. Therefore, the 1's in f are sufficient to fill h even if one takes into account that the 1 in cell (h, f) cannot be moved to (h, h). □

Lemma 4.2 *For each $f = 1, \ldots, n$, at iteration $n-f+1$ of the TOP-MOST-TIE algorithm, the column sums vector e of the maximal matrix $\bar{A}_f$ satisfies:*

$$e_1 \geq e_j, \quad j = 2, \ldots, f. \tag{19}$$

Proof We first observe that

$$r_1^*(C) \geq r_j^*(C) \quad \text{for } j = 2, \ldots, n.$$

Let $t \leq n$ be the number of rows i such that $r_i > 0$. We have $r_j^*(C) \leq t-1$, for each j, and $r_1^*(C) = t-1$. Since at the beginning of the algorithm $e_j = r_j^*(C)$ for each j, at the first iteration (19) holds.
Suppose now that (19) does not hold at iteration $n-f+1$, for some $f \leq n-1$. Then there exists a column k, $2 \leq k \leq f$, such that $e_k > e_1$. Since $\bar{A}_f$ is a maximal matrix, a 1 in column k implies a 1 in column 1 in all rows but row 1 for which we have $c_{11} = 0$. Thus, $e_k > e_1$ may hold only when $\bar{a}_{1k} = 1$ and $\bar{a}_{i1} = \bar{a}_{ik}, i = 2, \ldots, n$. Since $c_{kk} = 0$, it follows that $\bar{a}_{k1} = \bar{a}_{kk} = 0$, meaning that the 1 in cell $(k, 1)$ was already moved at a previous iteration under the condition that $\bar{r}_k = \max_{i=1,\ldots,,n} \bar{r}_i$. At that iteration, if $\bar{r}_k > 1$ the algorithm should have moved the rightmost 1 in row k instead of the 1 in cell $(k, 1)$; on the other hand, if $\bar{r}_k = 1$ the algorithm should have moved the 1 in cell $(1, k)$ instead of the 1 in cell $(k, 1)$. In any case we get a contradiction with the TOP-MOST-TIE rule. □

Theorem 4.3 *Let r and s be two nonnegative n-vectors defining an instance of 1-MFP for which hypothesis (6) and (10) hold. If $\mathcal{A}_C(r, s)$ is nonempty, the TOP-MOST-TIE algorithm correctly finds a $(0, 1)$-matrix in $\mathcal{A}_C(r, s)$.*

Proof We show that a contradiction arises whenever, being $\mathcal{A}_C(r, s)$ nonempty, one assumes that the algorithm prematurely stops at some iteration $n-f+1$ since it is not possible to satisfy the column sum s_f of column f. The latter situation may arise only in two cases: i) $e_f < s_f$ and $e_1 \leq s_f$; in this case, when filling column f one realizes that the 1's that can be moved from left to right in $\bar{A}_f$ are not sufficient to fill the whole matrix; ii) $e_f > s_f$; in this case, there is a surplus of 1's in column f that cannot be eliminated.

We notice that, when $e_f < s_f$ and $e_1 > s_f$, in column e_1 there are enough 1's that can be moved to column f. Thus, column f can be filled with s_f 1's even in the extreme case in which such 1's cannot be taken from columns 2 to $f-1$ in $\bar{A}_f$. It is clear that when $e_f = s_f$, column f already satisfies its total.

Suppose i) holds, i.e. $e_f < s_f$ and $e_1 \leq s_f$. By Lemma 4.2 and by (18) one has:

$$f e_1 \geq \sum_{j=1}^{f} e_j = \sum_{j=1}^{f} s_j.$$

From hypothesis (10) and $e_1 \leq s_f$ one has:

$$\sum_{j=1}^{f} s_j \geq f s_f \geq f e_1,$$

implying

$$f e_1 = \sum_{j=1}^{f} e_j = \sum_{j=1}^{f} s_j = f s_f.$$

Hence

$$e_1 = e_2 = \ldots = e_f = s_1 = s_2 = \ldots = s_f,$$

which contradicts $e_f < s_f$.

We now consider assumption ii), i.e., $e_f > s_f$. At iteration $n - f + 1$ column f is the next to be filled by the algorithm. We first show that, under assumption ii), up to this stage the algorithm had never moved 1's from columns $1, \ldots, f - 1$. Suppose on the contrary that at iteration $n - h + 1$, $h > f$, some 1's have been moved from a column $j < f$ to column h. By Lemma 4.1 this may happen only if

$$s_h \geq e_f^{n-h+1},$$

where e_f^{n-h+1} is the total number of 1's in column f at iteration $n - h + 1$. Since, during iterations $n - h + 1, \ldots, n - f + 2$, some 1's may have been moved from column f to the right, at iteration $n - f + 1$ one has:

$$e_f^{n-h+1} \geq e_f.$$

On the other hand, by assumption ii) and by the monotonicity of the column sum vector, one also has

$$e_f > s_f \geq s_h$$

leading to the contradiction $s_h > s_h$. Hence, no 1's were moved from columns $1, \ldots, f-1$ before iteration $n - f + 1$. It follows that, at the beginning of iteration $n - f + 1$, we have:

$$e_1 + e_2 + \ldots + e_{f-1} = r_1^*(C) + r_2^*(C) + \ldots + r_{f-1}^*(C). \tag{20}$$

By (18), (17) and (20), one has

$$\begin{aligned} e_1 + e_2 + \ldots + e_{f-1} + e_f &= s_1 + s_2 + \ldots + s_{f-1} + s_f \\ &\leq r_1^*(C) + r_2^*(C) + \ldots + r_{f-1}^*(C) + s_f \qquad (21) \\ &= e_1 + e_2 + \ldots + e_{f-1} + s_f \end{aligned}$$

thus implying $s_f \geq e_f$ which contradicts $e_f > s_f$.

□

To illustrate how the algorithm works, in the following we apply it to the 1-MPF instance of Examples 2. The displayed matrices correspond to the updates of $\bar{A}$ at the successive iterations of the algorithm. The last matrix is a feasible matrix in $\mathcal{A}_C(r, s)$. Notice that, according to (18), at iteration 4, when $f = 1$, the first column of $\bar{A}$ automatically satisfies its total $s_1 = 3$ and the algorithm stops with the feasible solution A:

$$\overset{\text{iteration } 0}{\begin{bmatrix} \times & 1 & 1 & 1 & 0 \\ 1 & \times & 1 & 0 & 0 \\ 1 & 1 & \times & 0 & 0 \\ 1 & 0 & 0 & \times & 0 \\ 1 & 0 & 0 & 0 & \times \end{bmatrix}} \quad \overset{\text{iteration } 1}{\begin{bmatrix} \times & 1 & 1 & 0 & 1 \\ 1 & \times & 1 & 0 & 0 \\ 1 & 1 & \times & 0 & 0 \\ 1 & 0 & 0 & \times & 0 \\ 1 & 0 & 0 & 0 & \times \end{bmatrix}} \quad \overset{\text{iteration } 2}{\begin{bmatrix} \times & 1 & 0 & 1 & 1 \\ 1 & \times & 1 & 0 & 0 \\ 1 & 1 & \times & 0 & 0 \\ 1 & 0 & 0 & \times & 0 \\ 1 & 0 & 0 & 0 & \times \end{bmatrix}}$$

$$\overset{\text{iteration 3}}{\begin{bmatrix} \times & 1 & 0 & 1 & 1 \\ 1 & \times & 1 & 0 & 0 \\ 1 & 1 & \times & 0 & 0 \\ 1 & 0 & 0 & \times & 0 \\ 1 & 0 & 0 & 0 & \times \end{bmatrix}} \qquad A = \overset{\text{iteration 4}}{\begin{bmatrix} \times & 1 & 0 & 1 & 1 \\ 1 & \times & 1 & 0 & 0 \\ 1 & 1 & \times & 0 & 0 \\ 0 & 1 & 0 & \times & 0 \\ 1 & 0 & 0 & 0 & \times \end{bmatrix}}.$$

5 Conclusions

In this paper we studied the Matrix Feasibility Problem (MFP), which was independently introduced by Gale and Ryser in 1957. For this problem both authors provided important results for the existence of a feasible solution and gave conceptually simple algorithms. Motivated by the elegance of these results, we started thinking about using Gale and Ryser algorithms for the solution of the Biproportional Apportionment Problem (BAP), a central problem in the theory of electoral systems. Actually, a straightforward application of these algorithms works well for this problem when each party receives votes in every electoral constituency, but some additional constraints must be taken into account when, for example, party j does not present its lists in constituency i at all. In this case, since it receives no votes, it cannot obtain any seat, due to the so called "zero-vote zero-seat" rule of BAP. This implies a fixed zero in cell (i, j) of the output matrix and additional constraints must be taken into account in the corresponding problem which then becomes a MFP "with fixed zeros". Looking at the possibility of extending the results by Gale and Ryser to this constrained version of MFP, we realized that, although this problem was already widely studied in the literature on combinatorial matrices, only few results exist for this case and they concern very particular configurations of the fixed zeros, while a general result does not exists.

In this paper we provide additional results for MFP on an $n \times n$ matrix, with $n = qp$ and fixed zeros in the p square submatrices of order q located on the main diagonal, a case that may arise in real applications of BAP. We also provide an algorithm for the special case $q = 1$ that is as simple as the one given by Ryser for the case without fixed zeros. Unfortunately, when fixed zeros are in arbitrary positions the problem is still open and additional effort is needed to reach a general result. Besides the theoretical importance of such a result, it must be pointed out that providing an algorithm that works in the general case would give rise to the possibility of producing a simple method for solving BAP which could be in fact acknowledged by the lawmakers and eventually written in an electoral law.

References

[1] R.K. Ahuja, T.L. Magnanti, J.B. Orlin. Network flows. Theory, algorithms and applications, Prentice Hall, New Jersey, 1993.

[2] R.P. Anstee (1982). Properties of a class of (0.1)-matrices covering a given matrix, *Canadian Journal of Mathematics*, 34, 438–453.

[3] M.L. Balinski, G. Demange (1989). An axiomatic approach to proportionality between matrices, *Mathematics of Operations Research*, 14, 700–719.

[4] M.L. Balinski, G. Demange (1989). Algorithms for proportional matrices in reals and integers, *Mathematical Programming*, 45, 193–210.

[5] R.A. Brualdi. Combinatorial Matrix Classes, Cambridge University Press, 2006.

[6] R.A. Brualdi, G. Dahl (2003). Matrices of zeros and ones with given line sums and a zero block, *Linear Algebra and its Applications*, 371, 191–207.

[7] R.A. Brualdi, G. Dahl (2007). Constructing $(0,1)$-Matrices with given line sums and certain fixed zeros. In Advances in Discrete Tomography and Its Applications, Eds.: Herman, G.T., Kuba, A., Birkhaser, Boston, 2007.

[8] L.H. Cox, L.R. Ernst (1982). Controlled rounding, *INFOR—Information Systems and Operational Research*, 20, 423–432.

[9] Y. Chen (2006). Simple existence conditions for zero-one matrices with at most one structural zero in each row and column, *Discrete Mathematics*, 306, 2870–2877.

[10] L.R. Ford, D.R. Fulkerson. Flows in networks, Princeton University Press, Princeton 1962.

[11] D.R. Fulkerson (1960). Zero-one matrices with zero trace, *Pacific Journal of Mathematics*, 10, 831–836.

[12] D. Gale (1957). A theorem on flows in networks, *Pacific Journal of Mathematics*, 7, 1073-1082.

[13] P. Grilli di Cortona, C. Manzi, A. Pennisi, F. Ricca, B. Simeone. Evaluation and Optimization of Electoral Systems, SIAM Monographs on Discrete Mathematics and Applications, SIAM, Society for Industrial and Applied Mathematics, Philadelphia 1999.

[14] G.H. Hardy, J.E. Littlewood, G. Pólya (1929). Some simple inequalities satisfied by convex functions, *Messenger Mathematics* 58, 145–152.

[15] Y. Nam (1999). Integral matrices with given row and column sums, *Ars Combinatoria* 52, 141–151.

[16] H.J. Ryser (1957). Combinatorial properties of matrices of zeros and ones, *Canadian Journal of Mathematics*, 9, 371–377.

[17] H.J. Ryser. Combinatorial Mathematics. Carus Math. Monograph ♯14, Math Assoc. of America, 1963.

[18] F. Ricca, A. Scozzari, P. Serafini, B. Simeone (2012), Error minimization methods in biproportional apportionment, TOP, The Official Journal of the Spanish Society of Statistics and Operations Research, in press, doi: 10.1007/s11750-012-0252-x.