

EFFICIENT GENERATION OF BINARY MAGIC SQUARES

Alain Riou

Sony Computer Science Laboratories – Paris
LTCI, Télécom Paris, Institut Polytechnique de Paris
alain.riou14000@yahoo.com

ABSTRACT

We propose a simple algorithm for generating Binary Magic Squares (BMS), i.e., square binary matrices where the sum of all rows and all columns are equal. We show by induction that our algorithm always returns valid BMS with optimal theoretical complexity. We then extend our study to non-square Binary Magic Squares, formalize conditions on the sum of rows and columns for these BMS to exist, and show that a slight variant of our first algorithm can generate provably generate them. Finally, we publicly release two implementations of our algorithm as Python packages, including one that can generate several BMS in parallel using GPU acceleration.

1 INTRODUCTION

Binary Magic Squares (BMS) are random binary matrices $M = (m_{ij})_{i,j} \in \{0,1\}^{n \times n}$ such that the sum of all rows and columns is equal to the same constant k , i.e.

$$\forall i \in \{0, \dots, n-1\}, \sum_{j=0}^{n-1} m_{ij} = \sum_{j=0}^{n-1} m_{ji} = k. \quad (1)$$

An example is provided on Figure 1.

BMS are strongly related to graph theory as they represent the adjacency matrix of regular bipartite graphs, leading to several applications, notably in combinatorial optimization [1].

Several works have theoretically studied the existence and computability of binary matrices with constraints on the sum of rows and columns, among which BMS are a special case [2, 3, 4]. A few works proposed to exhaustively generate all combinations of k -regular graphs [5]. Markov chain Monte Carlo algorithms for sampling binary matrices with fixed margins, typically relying on column swapping, have also been proposed [6, 7].

Here, we propose an efficient algorithm for generating BMS. We demonstrate that it always returns valid BMS with a complexity of $\mathcal{O}(n^2)$, and we then generalize it to non-square BMS. We release implementations of our algorithm as a Python pip-installable package.¹

0	1	1	0	1
0	1	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	1	0

Figure 1: A Binary Magic Square with $n = 5$ and $k = 3$.

¹<https://github.com/aRIOU/binary-magic-squares>

2 PRELIMINARIES

Before detailing our algorithm, we first observe that Binary Magic Squares always exist for any values of n and k .

Theorem 1. *For all $n \in \mathbb{N}$, for all $k \in \{0, \dots, n\}$, there exists $M \in \{0, 1\}^{n \times n}$ such that M is a valid BMS whose sum of rows and columns equals k .*

Proof. Let $n \in \mathbb{N}$ and $k \in \{0, \dots, n\}$. Consider the matrix M whose coefficients m_{ij} are defined by

$$m_{ij} = \begin{cases} 1 & \text{if } i \leq j < i + k \text{ or } i \leq j + n < i + k \\ 0 & \text{otherwise} \end{cases}. \quad (2)$$

For all $i \in \{0, \dots, n-1\}$,

$$\begin{aligned} \sum_{j=0}^{n-1} m_{ij} &= \sum_{j=i}^{\min(i+k-1, n-1)} m_{ij} + \sum_{j=0}^{i+k-n-1} m_{ij} \\ &= \min(i+k, n) - i + \max(i+k-n, 0) \\ &= k. \end{aligned} \quad (3)$$

Similarly, for all $j \in \{0, \dots, n-1\}$,

$$\begin{aligned} \sum_{i=0}^{n-1} m_{ij} &= \sum_{i=\max(j-k+1, 0)}^j m_{ij} + \sum_{i=j+n-k+1}^{n-1} m_{ij} \\ &= \min(j, k) + \max(k-j, 0) \\ &= k. \end{aligned} \quad (4)$$

By definition, M is therefore a Binary Magic Square. $\square$

This result, which is a special case of the Gale-Ryser theorem [8, 2], provides us the guarantee that our algorithm can always return a valid BMS whatever the input arguments.

3 EFFICIENT GENERATION OF BMS

3.1 INTUITION

The idea is to compute the BMS column by column. To do so, we start from a matrix $M \in \{0, 1\}^{n \times n}$ full of zeros and then, we successively pick k indices $(i_0, \dots, i_{k-1})$ per column t and put a 1 in the corresponding cells $(m_{i_l, t})_{0 \leq l < k}$, while perserving the following constraints on the sum over the rows through time:

$$\forall i \in \{0, \dots, n-1\}, \forall t \in \{0, \dots, n-1\}, \quad k + t - n \leq \sum_{j=0}^t m_{ij} \leq k. \quad (5)$$

If Equation (5) holds, in particular for $t = n-1$ the sum of each line i is $\sum_{j=0}^{n-1} m_{ij} = k$. Moreover, since we pick exactly k indices per column, the sum of each column is also k by construction, so M is actually a Binary Magic Square.

$s_0(t) = 3$	1	1	1	0	?	?	A_2
$s_1(t) = 2$	0	1	0	1	?	?	A_1
$s_2(t) = 4$	1	1	1	1	?	?	A_3
$s_3(t) = 2$	1	0	1	0	?	?	A_1
$s_4(t) = 2$	0	0	1	1	?	?	A_1
$s_5(t) = 3$	1	1	0	1	?	?	A_2

Figure 2: Illustration of our algorithm for a BMS of size 6×6 with $k = 4$. At the end of step t , we compute the current sum of each row. Then, indices are partitioned based on their value. At step $t + 1$, indices in A_1 must be selected to create a valid BMS, while indices from A_3 must not. Finally, some indices in A_2 are randomly selected (only one in this example) to ensure that the sum of each column equals k .

3.2 ALGORITHM

We now detail how to pick the indices at each step so that (5) is satisfied at each time step. The idea, illustrated in Figure 3.1, is that at each step t we partition the candidate indices into three subsets A_1 , A_2 and A_3 depending on whether the sum of the corresponding line is equal to $k + t - n$, equal to k or strictly in between, and pick the right indices accordingly.

Algorithm 1: Binary Magic Square generation

input : An integer $n \in \mathbb{N}$, an integer $k \in \{0, \dots, n\}$.

output: A Binary Magic Square $M \in \{0, 1\}^{n \times n}$ whose sum of rows and columns equals k .

```

1  $M = (m_{ij})_{i,j} = \mathbf{0}_{n \times n}$ ;
2 for  $i \in \{0, \dots, n-1\}$  do
3    $s_i = 0$ 
4 for  $t = 0$  to  $n-1$  do
5    $A_1 := \{i \in \{0, \dots, n-1\} \mid s_i = k + t - n\}$ ;
6    $A_2 := \{i \in \{0, \dots, n-1\} \mid k + t - n < s_i < k\}$ ;
7    $A_3 := \{i \in \{0, \dots, n-1\} \mid s_i = k\}$ ;
8    $E := A_1 \cup \text{random\_subset}(A_2, k - |A_1|)$ ;
9   for  $i \in E$  do
10     $m_{it} := 1$ ;
11     $s_i := s_i + 1$ ;
12 return  $M$ 

```

Here, E is the set of k indices that is picked at each time step. We can easily prove that at the end of each iteration t , we have

$$s_i = \sum_{j=0}^t m_{ij}. \quad (6)$$

4 CORRECTION AND COMPLEXITY

4.1 CORRECTION

Here, we demonstrate that Algorithm 1 returns valid BMS, first by checking the sum of rows remains equal, then the sum of the columns.

For each variable x in Algorithm 1, define $x(t)$ its t -th value in the algorithm.² In particular, the value of s_i may increase only 1 by 1, i.e. for all t

$$s_i(t) \leq s_i(t+1) \leq s_i(t) + 1. \quad (7)$$

Lemma 4.1. *The sum of each row of a matrix generated by Algorithm 1 equals k .*

Proof. We show by induction that Equation (5) is verified at each iteration t .

- $\forall i \in \{0, \dots, n-1\}, s_i(0) = 0$ so equation (5) holds for $t = 0$.
- Assume equation (5) holds for one $t \in \{0, \dots, n-1\}$. We show that it holds as well for $t+1$.

By hypothesis, for all $i \in \{0, \dots, n-1\}$, $k+t-n \leq s_i(t) \leq k$, so $(A_1(t), A_2(t), A_3(t))$ is a partition of $\{0, \dots, n-1\}$.

Then, for all $i \in \{0, \dots, n-1\}$,

$$\begin{aligned} - i \in A_1(t) & \quad i \in A_1(t) \Rightarrow s_i(t) = k+t-n \text{ and } i \in E(t) \\ & \quad \Rightarrow s_i(t+1) = s_i(t) + 1 = k+t-n+1 \\ - i \in A_2(t) & \quad i \in A_2(t) \Rightarrow k+t-n+1 \leq s_i(t) \leq k-1 \\ & \quad \Rightarrow k+t-n+1 \leq s_i(t+1) \leq k \quad \text{by (7)} \\ - i \in A_3(t) & \quad i \in A_3(t) \Rightarrow s_i(t) = k \text{ and } i \notin E(t) \\ & \quad \Rightarrow s_i(t+1) = s_i(t) = k \end{aligned}$$

So for all $i \in \{0, \dots, n-1\}$, $k+t-n+1 \leq s_i(t+1) \leq k$.

- By induction,

$$\forall t \in \{0, \dots, n-1\}, \forall i \in \{0, \dots, n-1\}, k+t-n \leq s_i(t) \leq k. \quad (8)$$

In particular, for all $i \in \{0, \dots, n-1\}$, $s_i(n) = k$, i.e. the sum of each row equals k . $\square$

Lemma 4.2. *The sum of each column of a matrix generated by Algorithm 1 equals k .*

Proof. For each row i and column t , $m_{it} = 1$ if $i \in E(t)$ and 0 otherwise, by design. Therefore for all t

$$\sum_{i=0}^{n-1} m_{it} = |E(t)|. \quad (9)$$

²Note that since s_i are defined before the **for** loop the sequence $(s_i(t))_t$ is defined up to $t = n$ whereas other variables are defined only up to $t = n-1$.

We show by induction that for all $t \in \{0, \dots, n-1\}$, $|E(t)| = k$.

- Assume $t = 0$.
 - If $k = 0$, then $A_1(0) = A_2(0) = \emptyset$ so $E(0) = \emptyset$ and $|E(0)| = 0 = k$.
 - If $k = n$, then $k + t - n = 0$, i.e. $A_1(0) = \{0, \dots, n-1\}$ and $A_2(0) = A_3(0) = \emptyset$. Then $E(0) = A_1(0) = \{0, \dots, n-1\}$ and $|E(0)| = n = k$.
 - If $0 < k < n$, then $A_1(0) = \emptyset$ and $A_2(0) = \{0, \dots, n-1\}$. Then, $E(0)$ is a subset of $A_2(0)$ of size $k - |A_1(0)|$, i.e. a subset of $\{0, \dots, n-1\}$ of size k . So $|E(0)| = k$.
- Assume that exists $0 < t \leq n-1$ such that for all $0 \leq j < t$, $|E(j)| = k$. We show that $|E(t)| = k$ as well.

At time step t , the total number of ones in M is then $\sum_{j=0}^{t-1} |E(j)| = tk$ (i.e. the sum of the ones in each column).

Also, the total number of ones in M can be expressed as the sum of the number of ones in each line, i.e. $\sum_{i=0}^{n-1} s_i(t)$.

As $A_1(t)$, $A_2(t)$ and $A_3(t)$ partition $\{0, \dots, n-1\}$, we have therefore

$$\begin{aligned} tk &= \sum_{i \in A_1(t)} s_i(t) + \sum_{i \in A_2(t)} s_i(t) + \sum_{i \in A_3(t)} s_i(t) \\ &= |A_1(t)|(k + t - n) + \sum_{i \in A_2(t)} s_i(t) + |A_3(t)|k \quad \text{by definition of } A_1 \text{ and } A_3 \end{aligned} \quad (10)$$

By definition of A_2 ,

$$|A_2(t)|(k + t - n) < \sum_{i \in A_2(t)} s_i(t) < |A_2(t)|k. \quad (11)$$

So, by injecting (11) into (10),

$$(|A_1(t)| + |A_2(t)|)(k + t - n) + |A_3(t)|k < tk < |A_1(t)|(k + t - n) + (|A_2(t)| + |A_3(t)|)k. \quad (12)$$

As $A_1(t)$, $A_2(t)$ and $A_3(t)$ partition $\{0, \dots, n-1\}$,

$$|A_1(t)| + |A_2(t)| + |A_3(t)| = n. \quad (13)$$

Therefore

$$\begin{aligned} (n - |A_3(t)|)(k + t - n) + |A_3(t)|k &< tk \\ nk + nt - n^2 - |A_3(t)|k - |A_3(t)|t + |A_3(t)|n + |A_3(t)|k &< tk \\ |A_3(t)|(n - t) &< n^2 - nk - nt + tk \\ |A_3(t)|(n - t) &< (n - k)(n - t) \\ |A_3(t)| &< n - k \end{aligned}$$

and

$$\begin{aligned} tk &< |A_1(t)|(k + t - n) + (n - |A_1(t)|)k \\ tk &< |A_1(t)|k + |A_1(t)|t - |A_1(t)|n + nk - |A_1(t)|k \\ |A_1(t)|(n - t) &< nk - tk \\ |A_1(t)| &< k \end{aligned}$$

Then, $E(t) = A_1(t) \cup \text{random_subset}(A_2(t), k - |A_1(t)|)$ by definition.

$$\begin{aligned} |A_2(t)| &= n - |A_1(t)| - |A_3(t)| \\ &> k - |A_1(t)| && \text{since } |A_3(t)| < n - k \\ &> 0 && \text{since } |A_1(t)| < k \end{aligned}$$

so $|\text{random_subset}(A_2(t), k - |A_1(t)|)| = k - |A_1(t)|$ and $|E(t)| = k$ since $A_1(t) \cap A_2(t) = \emptyset$.

- By induction, for all $t \in \{0, \dots, n-1\}$, $|E(t)| = k$.

□

Theorem 2. *Any matrix returned by Algorithm 1 is a Binary Magic Square.*

Proof. In Lemmas 4.1 and 4.2, we proved that the sum of every row and column of a matrix M generated by Algorithm 1 equals k . By definition, M is therefore a Binary Magic Square. □

4.2 COMPLEXITY

Without any parallelization trick, the overall complexity of Algorithm 1 is

$$C(n) = \mathcal{O}(n^2), \quad (14)$$

which is the optimal theoretical complexity for writing a $n \times n$ matrix.

However, all operations inside the **for** loop can be done using vectorized operations in practice. If we have p processes, the overall complexity of algorithm 1 then becomes

$$C(n) = \mathcal{O}\left(n \left\lceil \frac{n}{p} \right\rceil\right). \quad (15)$$

5 TOWARDS NON-SQUARE BINARY MAGIC SQUARES

5.1 CHARACTERIZATION

One can extend the definition of Binary Magic Squares to non-square matrices, by defining it as a matrix $M = (m_{ij})_{i,j} \in \{0, 1\}^{m \times n}$ such that

$$\begin{cases} \exists a \in \{0, \dots, n\}, \forall i \in \{0, \dots, m-1\}, \sum_{j=0}^{n-1} m_{ij} = a \\ \exists b \in \{0, \dots, m\}, \forall j \in \{0, \dots, n-1\}, \sum_{i=0}^{m-1} m_{ij} = b \end{cases}. \quad (16)$$

However, we show that not all combinations of a, b, m, n can lead to valid magic squares.

Theorem 3. *Let $m, n \in \mathbb{N}$, $a \in \{0, \dots, n\}$ and $b \in \{0, \dots, m\}$. There exists a valid non-square BMS $M \in \{0, 1\}^{m \times n}$ s.t. the sum of every row (resp. column) equals a (resp. b) iff the ratio between b and a equals the ratio between m and n , i.e., exists $q, q', m', n' \in \mathbb{N}$ such that $m = qm'$, $n = qn'$, $b = q'm'$ and $a = q'n'$.*

Proof. Let $M = (m_{ij})_{i,j} \in \{0, 1\}^{m \times n}$ be such a matrix. By definition,

$$\sum_{i=0}^{m-1} \sum_{j=0}^{n-1} m_{ij} = \sum_{j=0}^{n-1} \sum_{i=0}^{m-1} m_{ij}, \quad (17)$$

which leads to

$$am = bn. \quad (18)$$

Let $q = \gcd(m, n)$. There exists $m', n' \in \mathbb{N}$ such that $m = qm'$ and $n = qn'$, with $m' \wedge n' = 1$.

Unless $a = b = 0$, follows that $m' | bn'$ and $n' | am'$, hence $m' | b'$ and $n' | a$ by Euclid's lemma.

In other words, exists $q_a, q_b \in \mathbb{N}$ such that $a = q_a n'$ and $b = q_b m'$.

Equation (18) can be rewritten $q_a n' q m' = q_b m' q n'$, which ultimately leads to $q_a = q_b = q'$. Given $m, n \in \mathbb{N}$, the set of pairs (a, b) that can lead to valid BMS is therefore

$$\{(q'n', q'm') \mid q = \gcd(m, n), m = qm', n = qn', q' \in \{0, \dots, q\}\}. \quad (19)$$

We now demonstrate that under these conditions on a, b , there always exist a valid BMS.

Let $M = (m_{ij})_{i,j} \in \{0, 1\}^{q \times q}$ be a BMS whose sum of rows and columns equals q' (there exists at least one according to Theorem 1).

Then, we construct $M' \in \{0, 1\}^{m \times n}$ by repeating M m' times vertically and n' times horizontally. The sum of each row i of M equals $n' \sum_j m_{ij} = n' q' = a$. Similarly, the sum of each column j equals $m' \sum_i m_{ij} = m' q' = b$, which concludes the proof. $\square$

Remark. The condition on $\gcd(m, n)$ implies, in particular, that if $m \wedge n = 1$, then the only valid BMS are the trivial ones ($a = b = 0$ or $a = n$ and $b = m$).

5.2 ALGORITHM

The algorithm and proof of its correctness are very similar to those detailed above for square BMS. We include both here for completeness.

Algorithm 2: Non-square Binary Magic Square generation

input : Two integers $m, n \in \mathbb{N}$, two integers $a \in \{0, \dots, n\}$ and $b \in \{0, \dots, m\}$ satisfying the conditions of Theorem 3.

output: A Binary Magic Square $M \in \{0, 1\}^{m \times n}$ whose sum of rows and columns equals k .

```

1  $M = (m_{ij})_{i,j} = \mathbf{0}_{m \times n}$ ;
2 for  $i \in \{0, \dots, m-1\}$  do
3    $s_i = 0$ 
4 for  $t = 0$  to  $n-1$  do
5    $A_1 := \{i \in \{0, \dots, m-1\} \mid s_i = a + t - n\}$ ;
6    $A_2 := \{i \in \{0, \dots, m-1\} \mid a + t - n < s_i < a\}$ ;
7    $A_3 := \{i \in \{0, \dots, m-1\} \mid s_i = a\}$ ;
8    $E := A_1 \cup \text{random\_subset}(A_2, b - |A_1|)$ ;
9   for  $i \in E$  do
10     $m_{it} := 1$ ;
11     $s_i := s_i + 1$ ;
12 return  $M$ 

```

Lemma 5.1. The sum of each row of a matrix generated by Algorithm 2 equals a .

Proof. Similarly to the square case, here we show that for all $i \in \{0, \dots, m-1\}$, one has

$$a + t - n \leq s_i(t) \leq a. \quad (20)$$

We show by induction that Equation (20) is verified at each iteration t .

- $\forall i \in \{0, \dots, m-1\}, s_i(0) = 0$ so equation (20) holds for $t = 0$.
- Assume equation (5) holds for one $0 \leq t < n-1$. We show that it holds as well for $t+1$.
By hypothesis, for all $i \in \{0, \dots, m-1\}$, $a + t - n \leq s_i(t) \leq a$, so $(A_1(t), A_2(t), A_3(t))$ is a partition of $\{0, \dots, m-1\}$.

Then, for all $i \in \{0, \dots, m-1\}$,

$$\begin{aligned} - i \in A_1(t) & \quad i \in A_1(t) \Rightarrow s_i(t) = a + t - n \text{ and } i \in E(t) \\ & \quad \Rightarrow s_i(t+1) = s_i(t) + 1 = a + t - n + 1 \\ - i \in A_2(t) & \quad i \in A_2(t) \Rightarrow a + t - n + 1 \leq s_i(t) \leq a - 1 \\ & \quad \Rightarrow a + t - n + 1 \leq s_i(t+1) \leq a \quad \text{by equation (7)} \\ - i \in A_3(t) & \quad i \in A_3(t) \Rightarrow s_i(t) = a \text{ and } i \notin E(t) \\ & \quad \Rightarrow s_i(t+1) = s_i(t) = a \end{aligned}$$

So for all $i \in \{0, \dots, n-1\}$, $a + t - n + 1 \leq s_i(t+1) \leq a$.

- By induction,

$$\forall t \in \{0, \dots, n-1\}, \forall i \in \{0, \dots, m-1\}, a + t - n \leq s_i(t) \leq a. \quad (21)$$

In particular, for all $i \in \{0, \dots, m-1\}$, $s_i(n) = a$, i.e. the sum of each line equals a . $\square$

Lemma 5.2. *The sum of each column of a matrix generated by Algorithm 2 equals b .*

Proof. For each row i and column t , $m_{it} = 1$ if $i \in E(t)$ and 0 otherwise, by design. Therefore, for all t ,

$$\sum_{i=0}^{m-1} m_{it} = |E(t)|. \quad (22)$$

We show by induction that for all $t \in \{0, \dots, n-1\}$, $|E(t)| = b$.

- Assume $t = 0$.
 - If $b = 0$, then $a = 0$.
Consequently, $A_1(0) = A_2(0) = \emptyset$ so $E(0) = \emptyset$ and $|E(0)| = 0 = b$.
 - If $b = m$, then $a = n$, and $a + t - n = 0$, i.e. $A_1(0) = \{0, \dots, m-1\}$ and $A_2(0) = A_3(0) = \emptyset$.
Then $E(0) = A_1(0) = \{0, \dots, m-1\}$ and $|E(0)| = m = b$.

– If $0 < b < m$, then $0 < a < n$.

Consequently, $A_1(0) = \emptyset$ and $A_2(0) = \{0, \dots, m-1\}$.

Then, $E(0)$ is a subset of $A_2(0)$ of size $b - |A_1(0)|$, i.e. a subset of $\{0, \dots, m-1\}$ of size b . So $|E(0)| = b$.

- Assume that exists $0 < t \leq n-1$ such that for all $0 \leq j < t$, $|E(j)| = b$. We show that $|E(t)| = b$ as well.

At time step t , the total number of ones in M is then $\sum_{j=0}^{t-1} |E(j)| = tb$ (i.e. the sum of the ones in each column).

Also, the total number of ones in M can be expressed as the sum of the number of ones in each line, i.e. $\sum_{i=0}^{n-1} s_i(t)$.

As $A_1(t)$, $A_2(t)$ and $A_3(t)$ partition $\{0, \dots, m-1\}$, we have therefore

$$\begin{aligned} tb &= \sum_{i \in A_1(t)} s_i(t) + \sum_{i \in A_2(t)} s_i(t) + \sum_{i \in A_3(t)} s_i(t) \\ &= |A_1(t)|(a+t-n) + \sum_{i \in A_2(t)} s_i(t) + |A_3(t)|a \quad \text{by definition of } A_1 \text{ and } A_3 \end{aligned} \quad (23)$$

By definition of A_2 ,

$$|A_2(t)|(a+t-n) < \sum_{i \in A_2(t)} s_i(t) < |A_2(t)|a. \quad (24)$$

So, by injecting (24) into (23),

$$(|A_1(t)| + |A_2(t)|)(a+t-n) + |A_3(t)|a < tb < |A_1(t)|(a+t-n) + (|A_2(t)| + |A_3(t)|)a \quad (25)$$

As $A_1(t)$, $A_2(t)$ and $A_3(t)$ partition $\{0, \dots, m-1\}$,

$$|A_1(t)| + |A_2(t)| + |A_3(t)| = m. \quad (26)$$

Therefore

$$\begin{aligned} (m - |A_3(t)|)(a+t-n) + |A_3(t)|a &< tb \\ am + mt - mn - |A_3(t)|a - |A_3(t)|t + |A_3(t)|n + |A_3(t)|a &< tb \\ |A_3(t)|(n-t) &< mn - am - mt + tb \\ |A_3(t)|(n-t) &< mn - bn - mt + tb && \text{by (18)} \\ |A_3(t)|(n-t) &< (m-b)(n-t) \\ |A_3(t)| &< m-b \end{aligned}$$

and

$$\begin{aligned} tb &< |A_1(t)|(a+t-n) + (m - |A_1(t)|)a \\ tb &< |A_1(t)|a + |A_1(t)|t - |A_1(t)|n + am - |A_1(t)|a \\ |A_1(t)|(n-t) &< am - tb \\ |A_1(t)|(n-t) &< bn - tb && \text{by (18)} \\ |A_1(t)| &< b \end{aligned}$$

Then, $E(t) = A_1(t) \cup \text{random_subset}(A_2(t), b - |A_1(t)|)$ by definition.

$$\begin{aligned} |A_2(t)| &= m - |A_1(t)| - |A_3(t)| \\ &> b - |A_1(t)| && \text{since } |A_3(t)| < m - b \\ &> 0 && \text{since } |A_1(t)| < b \end{aligned}$$

so $|\text{random_subset}(A_2(t), b - |A_1(t)|)| = b - |A_1(t)|$ and $|E(t)| = b$ since $A_1(t) \cap A_2(t) = \emptyset$.

- By induction, for all $t \in \{0, \dots, n-1\}$, $|E(t)| = b$.

□

Theorem 4. *Any matrix returned by Algorithm 2 is a non-square Binary Magic Square.*

Proof. In Lemmas 5.1 and 5.2, we proved that the sum of every row (resp. column) of a matrix M generated by Algorithm 2 equals a (resp. b). By definition, M is therefore a Binary Magic Square. □

6 CONCLUSION

We introduced an efficient algorithm and implementation for generating random Binary Magic Squares, and showed that it can easily be generalized to non-square matrices. Binary matrices with constraints on the sums of rows and columns have been widely studied, however simple algorithms with open implementations targeting the specific case of Binary Magic Squares were, to the best of our knowledge, lacking. Here, we therefore propose such an algorithm, theoretically validate its correctness, and publicly release a Python implementation. In particular, we also propose a PyTorch version of our algorithm which can generate many BMS in parallel on a GPU.

REFERENCES

- [1] W. K. Chen, “Graph theory and its engineering applications,” *Graph Theory and Its Engineering Applications*, 2 1997.
- [2] H. J. Ryser, “Combinatorial properties of matrices of zeros and ones,” *Canadian Journal of Mathematics*, vol. 9, pp. 371–377, 1957.
- [3] Y. Nam, “Integral matrices with given row and column sums,” *Ars Combinatoria*, vol. Volume 052, pp. 141–151, 1999.
- [4] A. Alpers and P. Gritzmann, “Reconstructing binary matrices under window constraints from their row and column sums,”
- [5] M. Meringer, “Fast generation of regular graphs and construction of cages,” 1999.
- [6] G. Wang, “A fast mcmc for the uniform sampling of binary matrices with fixed margins,” 2019.
- [7] A. Fout, B. K. Fosdick, and M. P. Hitt, “Non-uniform sampling of fixed margin binary matrices,” *FODS 2020 - Proceedings of the 2020 ACM-IMS Foundations of Data Science Conference*, pp. 95–105, 10 2020.
- [8] D. Gale, “A theorem on flows in networks,” *Pacific Journal of Mathematics*, vol. 7, pp. 1073–1082, 1957.