AUSTRALASIAN JOURNAL OF COMBINATORICS
Volume **69(1)** (2017), Pages 130–144

Computational results on invertible matrices with the maximum number of invertible 2×2 submatrices

Navid Nasr Esfahani Douglas R. Stinson*

David R. Cheriton School of Computer Science
University of Waterloo
Waterloo, Ontario N2L 3G1
Canada

Abstract

A linear 2-All-or-Nothing Transform can be considered as an invertible matrix with all 2×2 submatrices invertible. It is known [P. D'Arco, N. Nasr Esfahani and D.R. Stinson, *Electron. J. Combin.* 23(4) (2016), #P4.10] that there is no binary $s \times s$ matrix that satisfies these conditions, for $s > 2$. In this paper, different computational methods for generating invertible binary matrices with close to the maximum number of invertible 2×2 submatrices have been implemented and compared against each other. We also study the ternary matrices with such properties.

1 Introduction

All-or-nothing transforms were first introduced by Rivest [7] as a mode of operation, for block ciphers, that slows down brute force attacks proportionally to the number of blocks. To achieve that, he defines an *all-or-nothing transform* (AONT) as a reversible transformation, from message blocks to output blocks, that both itself and its inverse are "efficiently computable", and obtaining any information about any message block without the knowledge of all output blocks is "computationally infeasible". Stinson [8] extended the definition and introduced unconditionally secure AONTs. Since this paper uses the unconditionally secure AONTs, we will start with their definition.

Definition 1.1. *Let Σ be a finite set of size q. For a positive integer s, $\phi : \Sigma^s \to \Sigma^s$ is an unconditionally secure all-or-nothing transform if*

1. *ϕ is a bijection.*

* D. Stinson's research is supported by NSERC discovery grant #RGPIN-03882.

2. *Given any output subset* $\mathcal{Y} \subset \{y_1, y_2, \ldots, y_s\}$ *where* $|\mathcal{Y}| = s-1$*, no information about any single input element* $x_i, 1 \leq i \leq s$ *can be obtained.*

Stinson [8] also focuses on *linear AONTs*, which are transforms that can be represented by invertible matrices, and studies Hadamard matrices and orthogonal arrays as instances of such AONTs. Besides the initial application, AONTs have been used in exposure-resilient functions [1], network coding [2, 5, 9], anti-jamming techniques [6], query anonymization[11], etc.

D'Arco et al. [3] discuss the information theoretical security of more than one input block and more specifically focus on the case of two input blocks. Following from Definition 1.1, t-AONTs are defined as below.

Definition 1.2. *Let* Σ *be a finite set of size* q*. For a positive integer* s*,* $\phi : \Sigma^s \to \Sigma^s$ *is an unconditionally secure* t*-all-or-nothing transform if*

1. ϕ *is a bijection.*

2. *Given any output subset* $\mathcal{Y} \subset \{y_1, y_2, \ldots, y_s\}$ *where* $|\mathcal{Y}| = s-t$*, no information about any input subset* $\mathcal{X} \subset \{x_1, x_2, \ldots, x_s\}, |\mathcal{X}| = t$ *can be obtained.*

The authors prove that for $s > 2$, there is no binary 2-AONT, and thus they define $N_2(s)$ and $R_2(s)$ as measures for how "close" a transform can be to an $s \times s$ 2-AONT. $N_t(s)$ is the maximum possible number of invertible $t \times t$ submatrices of an invertible $s \times s$ matrix, and

$$R_t(s) = \frac{N_t(s)}{\binom{s}{t}^2}.$$

Then they continue by presenting different constructions, as well as an upper bound on maximum observed values of $R_2(s)$ based on quadratic programming. Zhang et al. [12] further studied the $t = 2$ case and showed that $R_2(s)$ converges to 0.5 as s increases. In this paper, we will denote N_t^s in conjunction with an algorithm as the maximum number of $t \times t$ submatrices found by the algorithm, e.g. "Cyclic N_2^s" refers to the maximum number of invertible 2×2 submatrices in an invertible $s \times s$ matrix found by the algorithm which searches the cyclic matrices, and

$$R_t^s = \frac{N_t^s}{\binom{s}{t}^2}.$$

It follows from the definitions that "Exhaustive N_t^s" is equal to $N_t(s)$.

As mentioned above, in addition to the analytical work on invertible matrices with the maximum number of invertible 2×2 submatrices, different constructions have been presented based on balanced incomplete block designs [3], cyclotomy [3, 12], random constructions [3], solving integer-programming models [12], and exhaustive search [3]. In this paper, three search algorithms for transforms "close to" AONTs are presented, as well as the computational results of those algorithms. First, the exhaustive search used in [3] is explained, then the exhaustive search on cyclic matrices is considered, and finally we discuss a search for matrices that are one entry off from being cyclic.

2 Search Algorithms

D'Arco et al. [3] presented a quadratic programming (QP) formulation for 2-AONTs, given below:

$$\begin{array}{|ll|}\hline \text{Maximize} & \frac{1}{2}\mathbf{z}C\mathbf{z}^T \\ \text{subject to} & \sum_{i=1}^{15} z_i \leq 1, \\ & z_i \geq 0,\ 1 \leq i \leq 15. \\ \hline \end{array}$$

The matrix C is defined such that c_{ij} is the number of invertible 2×2 submatrices formed by considering binary representations of i and j as two rows. Thus, the main diagonal of C is all zeros. Since the trace of C equals 0 and C is not an all-zero matrix, it has both positive and negative eigenvalues. Therefore, the matrix C is not positive/negative semi-definite for any value of s. According to Vavasis [10, p. 81], the above-mentioned QP problem is NP-hard for such matrices. Consequently, an exhaustive search algorithm was used to search for an instance of invertible $s \times s$ matrices with the maximum possible number of invertible 2×2 submatrices, for $4 \leq s \leq 9$. The algorithm used a branch-and-bound technique that branches by iterating over different possible combinations for each row, in nested loops, and bounds the search as soon as the rows of the matrix become linearly dependent.

Based on the idea of constructing matrices "close to" 2-AONTs using cyclotomy in [3] and [12], and also due to computational limitations of enumerating all possible cases of matrices, we decided to limit the search to cyclic matrices. The algorithm for this search iterates over different possible values for the first row of the matrix, and each of the other rows will be generated by applying a cyclic shift to the row above.

Although cyclic matrices give optimum or near optimum results in most of the cases that are possible to check, the case $s = 3$ illustrates their weakness. In this case, going from a cyclic matrix to a matrix that is only one element away from being cyclic increases the ratio from $0.\bar{3}$ to $0.\bar{7}$. This improvement encouraged us to consider applying the "adjusting step" suggested by Zhang et al. [12].

These search algorithms will be discussed in detail in the following subsections.

2.1 Exhaustive Search

In total, there are 2^{n^2} different binary $n \times n$ matrices; however many of them can be skipped because either they are not invertible, or a permutation of their rows and columns has already been considered. In the search algorithm, each `for` loop iterates over the possible values for a row. At each iteration, if a row is a linear combination of the rows above it, that row will not be considered, i.e., the search will be bounded by this linear dependency check. Besides, since any permutation of rows and columns does not effect either the singularity, or the number of invertible 2×2 submatrices of a matrix, we want to enumerate as few matrices of each class as possible. Therefore, the search algorithm only generated matrices in which each

row has at least as many 1's as the number of 1's in the row above it. Also, if two rows have the same number of 1's, the row representing a smaller number in binary, should appear higher. These two rules enabled us to search only a $\frac{1}{s!}$ fraction of the search domain. Finally, we partially restricted column permutations by fixing all the 1's in the first row to be the right-most coordinates. This constraint helped the algorithm to skip repeated computations over different permutations of the first row, for a fixed number of 1's.

Example 2.1. *Let $s = 4$. Then following the bounding constraints in the exhaustive algorithm search, the first row can only be chosen from:* $(0,0,0,1), (0,0,1,1)$, $(0,1,1,1), (1,1,1,1)$. *Suppose (0,0,1,1) is the chosen first row, then neither* $(0,0,0,1)$, $(0,0,1,0), (0,1,0,0)$, *nor* $(1,0,0,0)$ *can be chosen as the second row because their weights are smaller than that of the first row. Now, if the second row is* $(1,0,0,1)$, *then the third row cannot be* $(0,1,1,0)$ *as it represents 6 in binary which is smaller than 9, which is represented by the second row; also, it cannot be* $(1,0,1,0)$ *because it is a linear combination of the first two rows.*

Another restriction was added for the case when $s = 9$: for the first 5 rows, if two coordinates in a row have different values, but in all the rows above them, those coordinates were identical, in this row the value 0 should appear on the left side of the value 1.

The computations for $4 \leq s \leq 8$ were executed on one node on a server of the Cheriton School of Computer Science, `linux.cs.uwaterloo.ca`, with a 64 bit AMD CPU, having a 2.6 GHz clock rate. We also attempted to use the same algorithm distributed over 256 processors on `grex.westgrid.ca`. But 14 of those processes did not terminate by the end of the 96 hour time limit. The search domain, corresponding to those processes, was distributed again among 266 processes. In total, the whole computation took approximately 10000 CPU hours.

Some information about the resultant matrices is provided below. Also, the pseudocode in Algorithm 1 illustrates the general algorithm used in the processes. For the $s = 9$ case, different iterations of the second `for` loop were distributed among 256 processes, two iterations per process.

Table 1 presents the number of 1's, their density, and minimum weight of a row in the resultant matrices, along with the values for s, $N_2(s)$, and $R_2(s)$.

Algorithm 1 Exhaustive search

```
1: currSoln: an array of length s (Stores the solution under consideration)
2: soln: an array of length s (Stores the best solution so far)
3: Mats: an array of length s (Stores the number of invertible 2 × 2 matrices up to
   this level)
4: weight: an array of length 2^s (Stores the number of ones in the binary expansion
   of its index plus one)
5: for î : 1 → s − 1 do
6:    i : 2^î − 1
7:    currSoln[0] ← i + 1 (+1 because indices start from 0)
8:    for j : i + 1 → 2^s − s + 2 (−s + 2 because these many more rows are needed)
      do
9:       if weight[i] ≤ weight[j] then
10:         currSoln[1] ← j
11:         Mats[1] ← C[i][j]
12:         for k : j + 1 → 2^s − s + 3 do
13:            if weight[j] ≤ weight[k] then
14:               currSoln[2] ← k
15:               Mats[2] ← Mats[1] + C[i][k] + C[j][k]
16:               if Chosen rows are linearly independent then
17:                  ...
18:                  for ℓ : prev. counter + 1 → 2^s do
19:                     if weight[prev. counter] ≤ weight[ℓ] then
20:                        currSoln[s − 1] ← ℓ + 1
21:                        Mats[s − 1] ← Mats[s − 2] + C[i][ℓ] + C[j][ℓ] + ...
22:                        if Mats[s − 1] > max AND Chosen rows are linearly inde-
                           pendent then
23:                           max ← Mats[s − 1]
24:                           soln ← currSoln
25:                        end if
26:                     end if
27:                  end for
28:                  ...
29:               end if
30:            end if
31:         end for
32:      end if
33:   end for
34: end for
35: return soln
```

s	$N_2(s)$	$R_2(s)$	1's in the matrix	density of 1's	Min.weight of a row
3	7	$0.\bar{7}$	7	$0.\bar{7}$	2
4	30	$0.8\bar{3}$	12	0.75	3
5	70	0.7	17	0.68	3
6	150	$0.\bar{6}$	25	$0.69\bar{4}$	4
7	287	≈ 0.651	35	≈ 0.714	5
8	485	≈ 0.618	47	≈ 0.734	5
9	783	≈ 0.604	55	≈ 0.679	6

Table 1: Number and density of 1's in the invertible matrices with maximum number of invertible 2×2 submatrices for $s = 3, \ldots, 9$.

2.2 Search for Cyclic Matrices

In order to search all the invertible $s \times s$ cyclic matrices for the one with the greatest $R_2(s)$ value, it suffices to iterate through all possible combinations for the first row. This is because the order of rows does not affect the value of $R_2(s)$. It also guarantees that the first row can be substituted by any of its cyclic shifts. Hence, only one representative of each class of rows, resulting by shifting the first row, need to be examined. For each choice of the first row, the number of invertible 2×2 submatrices generated by that row and any of its shifts are counted and multiplied by $\frac{s}{2}$, in order to compute the total number of invertible 2×2 submatrices for that cyclic matrix. The algorithm keeps the row resulting in the best ratio found so far, and reports that row at the end of the search.

The cyclic search program, for $3 \leq s \leq 36$, was sequentially executed on a node on `linux.cs.uwaterloo.ca` in about 14 hours. Table 2 demonstrates the results of the cyclic search and the exhaustive search, as far as possible, for the sake of comparison.

As Table 2 shows, $s = 2$, $s = 4$, and $s = 7$ are the only cases, as far as we can compare, where both algorithms generate similar results.

2.3 Search for Almost Cyclic Matrices

As previously mentioned, a cyclic search may fail to find some solutions near optimal solution. This limitation can be attributed to restriction on the matrices to be cyclic. Since being cyclic is not an intrinsic property of AONTs, the condition can be relaxed so that the search considers matrices that are almost cyclic and have large 2-density. To do so, we developed a modification of the adjusting step by Zhang et al. [12]. The algorithm searches the matrices that are cyclic or off-by-one from being cyclic, i.e., a cyclic matrix with one entry altered from 0 to 1 or from 1 to 0, from the matrix with the maximum 2-density.

s	1's / row	1 Frq	Max Cyc N_2	Max Cyc R_2	$N_2(s)$	$R_2(s)$
2	1	0.5	1	1	1	1
3	1	$0.\bar{3}$	3	$0.\bar{3}$	7	$0.\bar{7}$
4	3	0.75	30	$0.8\bar{3}$	30	$0.8\bar{3}$
5	3	0.6	65	0.65	70	0.7
6	5	$0.8\bar{3}$	135	0.6	150	$0.\bar{6}$
7	5	≈ 0.714	287	≈ 0.651	287	≈ 0.651
8	5	0.625	468	≈ 0.597	485	≈ 0.619
9	7	$0.\bar{7}$	765	≈ 0.590	783	≈ 0.604
10	7	0.7	1215	0.6	–	–
11	7	≈ 0.636	1716	≈ 0.567	–	–
12	9	0.75	2502	≈ 0.574	–	–
13	9	≈ 0.692	3510	≈ 0.577	–	–
14	9	≈ 0.643	4557	≈ 0.550	–	–
15	11	$0.7\bar{3}$	6210	≈ 0.563	–	–
16	11	≈ 0.688	8040	≈ 0.558	–	–
17	13	≈ 0.765	10030	≈ 0.542	–	–
18	13	$0.7\bar{2}$	12933	≈ 0.552	–	–
19	13	≈ 0.684	16017	≈ 0.548	–	–
20	15	0.75	19510	≈ 0.540	–	–
21	15	≈ 0.714	24045	≈ 0.545	–	–
22	15	≈ 0.681	28831	≈ 0.540	–	–
23	17	≈ 0.739	34385	≈ 0.537	–	–
24	17	≈ 0.708	41124	≈ 0.540	–	–
25	17	0.68	48100	≈ 0.534	–	–
26	19	≈ 0.731	56433	≈ 0.534	–	–
27	19	≈ 0.704	65934	≈ 0.535	–	–
28	19	≈ 0.679	75726	≈ 0.530	–	–
29	21	≈ 0.724	87638	≈ 0.532	–	–
30	21	0.7	100485	≈ 0.531	–	–
31	21	≈ 0.677	113863	≈ 0.527	–	–
32	23	≈ 0.719	130128	≈ 0.529	–	–
33	23	$0.6\bar{9}$	147213	≈ 0.528	–	–
34	25	≈ 0.735	165087	≈ 0.525	–	–
35	25	≈ 0.714	186445	≈ 0.527	–	–
36	25	≈ 0.694	208530	≈ 0.525	–	–

Table 2: Number of ones in a row of cyclic matrices, and density of ones in them are provided. Also the number of invertible 2×2 submatrices and their ratio to the total number of all 2×2 submatrices in cyclic matrices can be compared with those values in maximum cases for $s = 2, \ldots, 8$.

s	Cyc N_2	Cyc R_2	Adj Cyc N_2	Adj Cyc R_2
2	1	1	1	1
3	3	$0.\bar{3}$	7	$0.\bar{7}$
4	30	$0.8\bar{3}$	30	$0.8\bar{3}$
5	65	0.65	69	0.69
6	135	0.6	148	≈ 0.658
7	287	≈ 0.651	287	≈ 0.651
8	468	≈ 0.597	485	≈ 0.619
9	765	≈ 0.590	781	≈ 0.603
10	1215	0.6	1215	0.6
11	1716	≈ 0.567	1777	≈ 0.587
12	2502	≈ 0.574	2503	≈ 0.575
13	3510	≈ 0.577	3510	≈ 0.577
14	4557	≈ 0.550	4707	≈ 0.568
15	6210	≈ 0.563	6210	≈ 0.563
16	8040	≈ 0.558	8040	≈ 0.558
17	10030	≈ 0.542	10288	≈ 0.556
18	12933	≈ 0.552	12933	≈ 0.552
19	16017	≈ 0.548	16017	≈ 0.548
20	19510	≈ 0.540	19746	≈ 0.547
21	24045	≈ 0.545	24045	≈ 0.545
22	28831	≈ 0.540	28905	≈ 0.542
23	34385	≈ 0.537	34584	≈ 0.540
24	41124	≈ 0.540	41124	≈ 0.540
25	48100	≈ 0.534	48364	≈ 0.537
26	56433	≈ 0.534	56544	≈ 0.535
27	65934	≈ 0.535	65934	≈ 0.535
28	75726	≈ 0.530	76225	≈ 0.533
29	87638	≈ 0.532	87638*	≈ 0.537*
30	100485	≈ 0.531	100485*	≈ 0.531*
31	113863	≈ 0.527	114642*	≈ 0.530*
32	130128	≈ 0.529	130128*	≈ 0.529*
33	147213	≈ 0.528	147213*	≈ 0.528*
34	165087	≈ 0.525	166026*	≈ 0.528*
35	186445	≈ 0.527	186445*	≈ 0.527*
36	208530	≈ 0.525	208530*	≈ 0.525*

Table 3: Comparison of N_2 and R_2 values for cyclic matrices and matrices with one element adjustment for $s = 2, \ldots, 36$ and the maximum of number of 2×2 submatrices possible in a binary cyclic matrix as an upper bound for $q = 2$. For $29 \leq s \leq 36$ the adjusting step algorithm was unable to find an answer, thus a simpler modification of the algorithm was used, which does not check all the matrices that are one entry away from being cyclic. The results from the modified algorithm are marked with $*$.

The search algorithm enumerates the matrices in the same method that the cyclic search does; however, it does not consider the independence of the rows as a necessary condition if the rank of the resultant matrix is $n-1$. Instead, the algorithm tries flipping each of the entries in the last row of the matrix, one at a time, and checks the independence of the rows and the number of invertible 2×2 submatrices of each of the new matrices. The 2-densities of these matrices are then compared, first among themselves and then to best 2-density found so far. It should be noted that any entry can be moved to the last row without changing the values of the other entries through cyclic shifts of the rows[1] followed by cyclic shifts of the columns; therefore, it is sufficient to flip entries only in the last row of the matrix because the matrix is cyclic.

The computations were executed sequentially, on a node on `linux.cs.uwaterloo.ca`, in about 16 hours.

Table 3 compares the results of cyclic search to those of cyclic search with adjusting step. It can be seen from the table that the adjusting step improves the results in 18 out of 35 cases, and for the case $s = 8$ the algorithm performs as well as the exhaustive search. The rate of improvement is the most significant for $s = 3$, where the adjusting step improves the result by more than 130%. This rate decreases to less than 1% for $s = 34$, where flipping one entry increases the number of invertible 2×2 submatrices by 939.

3 Invertible Matrices with $q = 3$ Symbols

3.1 Random Construction

In this section we consider invertible matrices with entries from $\{0, 1, 2\}$. First, we will consider the random construction of such matrices. In this case, there are $(3^2-1)(3^2-3) = 48$ invertible 2×2 matrices, as listed below.

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}, \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}, \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}, \begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix},$$

$$\begin{pmatrix} 0 & 2 \\ 2 & 0 \end{pmatrix}, \begin{pmatrix} 2 & 2 \\ 2 & 0 \end{pmatrix}, \begin{pmatrix} 2 & 2 \\ 0 & 2 \end{pmatrix}, \begin{pmatrix} 2 & 0 \\ 2 & 2 \end{pmatrix}, \begin{pmatrix} 0 & 2 \\ 2 & 2 \end{pmatrix}, \begin{pmatrix} 1 & 1 \\ 1 & 2 \end{pmatrix}, \begin{pmatrix} 1 & 1 \\ 2 & 1 \end{pmatrix},$$

$$\begin{pmatrix} 1 & 2 \\ 1 & 1 \end{pmatrix}, \begin{pmatrix} 2 & 1 \\ 1 & 1 \end{pmatrix}, \begin{pmatrix} 2 & 2 \\ 2 & 1 \end{pmatrix}, \begin{pmatrix} 2 & 2 \\ 1 & 2 \end{pmatrix}, \begin{pmatrix} 2 & 1 \\ 2 & 2 \end{pmatrix}, \begin{pmatrix} 1 & 2 \\ 2 & 2 \end{pmatrix}, \begin{pmatrix} 1 & 0 \\ 0 & 2 \end{pmatrix},$$

$$\begin{pmatrix} 0 & 1 \\ 2 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 2 \\ 1 & 0 \end{pmatrix}, \begin{pmatrix} 2 & 0 \\ 0 & 1 \end{pmatrix}, \begin{pmatrix} 0 & 1 \\ 1 & 2 \end{pmatrix}, \begin{pmatrix} 0 & 1 \\ 2 & 1 \end{pmatrix}, \begin{pmatrix} 0 & 2 \\ 1 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 1 \\ 0 & 2 \end{pmatrix},$$

$$\begin{pmatrix} 1 & 1 \\ 2 & 0 \end{pmatrix}, \begin{pmatrix} 1 & 0 \\ 1 & 2 \end{pmatrix}, \begin{pmatrix} 1 & 0 \\ 2 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 2 \\ 0 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 2 \\ 1 & 0 \end{pmatrix}, \begin{pmatrix} 2 & 0 \\ 1 & 1 \end{pmatrix}, \begin{pmatrix} 2 & 1 \\ 0 & 1 \end{pmatrix},$$

[1] Rows and columns are shifted as a whole, not the entries in them

$$\begin{pmatrix} 2 & 1 \\ 1 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 2 \\ 2 & 1 \end{pmatrix}, \begin{pmatrix} 0 & 2 \\ 1 & 2 \end{pmatrix}, \begin{pmatrix} 0 & 1 \\ 2 & 2 \end{pmatrix}, \begin{pmatrix} 2 & 2 \\ 0 & 1 \end{pmatrix}, \begin{pmatrix} 2 & 2 \\ 1 & 0 \end{pmatrix}, \begin{pmatrix} 2 & 0 \\ 2 & 1 \end{pmatrix},$$
$$\begin{pmatrix} 2 & 0 \\ 1 & 2 \end{pmatrix}, \begin{pmatrix} 2 & 1 \\ 0 & 2 \end{pmatrix}, \begin{pmatrix} 2 & 1 \\ 2 & 0 \end{pmatrix}, \begin{pmatrix} 1 & 0 \\ 2 & 2 \end{pmatrix}, \begin{pmatrix} 1 & 2 \\ 0 & 2 \end{pmatrix}, \begin{pmatrix} 1 & 2 \\ 2 & 0 \end{pmatrix}$$

A random 2×2 matrix $A = \{a_{i,j}\}$ that is generated by randomly assigning its entries to 0, 1, and 2, with the following probabilities

$$Pr(a_{i,j} = 1) = \alpha, Pr(a_{i,j} = 2) = \beta, Pr(a_{i,j} = 0) = \gamma, \alpha + \beta + \gamma = 1$$

is therefore invertible with probability

$$\begin{aligned} f(\alpha, \beta, \gamma) = 2(\alpha^2\gamma^2 + \beta^2\gamma^2) + 4(\alpha^3\gamma + \beta^3\gamma) \\ +4(\alpha^3\beta + \beta^3\alpha) + 12(\alpha^2\beta\gamma + \beta^2\alpha\gamma) + 4\alpha\beta\gamma^2. \end{aligned} \tag{1}$$

To maximize f we use the Lagrange multiplier method. First, define g as follows:

$$\begin{aligned} g(\alpha, \beta, \gamma, \lambda) = f(\alpha, \beta, \gamma) - \lambda(1 - \alpha - \beta - \gamma) = 2(\alpha^2\gamma^2 + \beta^2\gamma^2) \\ +4(\alpha^3\gamma + \beta^3\gamma) + 4(\alpha^3\beta + \beta^3\alpha) + 12(\alpha^2\beta\gamma + \beta^2\alpha\gamma) + 4\alpha\beta\gamma^2 \\ -\lambda(1 - \alpha - \beta - \gamma). \end{aligned} \tag{2}$$

Computing all four partial derivatives of g, we get

$$\frac{\partial g}{\partial \alpha} = 4\alpha\gamma^2 + 12\alpha^2\gamma + 12\alpha^2\beta + 4\beta^3 + 24\alpha\beta\gamma + 12\beta^2\gamma + 4\beta\gamma^2 - \lambda \tag{3}$$

$$\frac{\partial g}{\partial \beta} = 4\beta\gamma^2 + 12\beta^\gamma + 12\beta^2\alpha + 4\alpha^3 + 24\alpha\beta\gamma + 12\alpha^2\gamma + 4\alpha\gamma^2 - \lambda \tag{4}$$

$$\frac{\partial g}{\partial \gamma} = 4\alpha^2\gamma + 4\beta^2\gamma + 4\alpha^3 + 4\beta^3 + 12\alpha^2\beta + 12\beta^2\alpha + 8\alpha\beta\gamma - \lambda \tag{5}$$

$$\frac{\partial g}{\partial \lambda} = -\alpha - \beta - \gamma + 1 = 0. \tag{6}$$

We used Maple to solve $\frac{\partial g}{\partial \alpha} = 0, \frac{\partial g}{\partial \beta} = 0, \frac{\partial g}{\partial \gamma} = 0, \frac{\partial g}{\partial \lambda} = 0$, and the only solution that maximizes g, and therefore f, and which satisfies $\alpha, \beta, \gamma \in [0, 1]$ is $\alpha = \beta = \frac{1}{\sqrt{6}}, \gamma = 1 - \frac{2}{\sqrt{6}}, \lambda = \frac{8}{3}$, from which we obtain $g(\frac{1}{\sqrt{6}}, \frac{1}{\sqrt{6}}, 1 - \frac{2}{\sqrt{6}}, \frac{8}{3}) = \frac{2}{3}$.

For each value of s, $3 \leq s \leq 13$, 10000 random matrices with the distribution above were generated. For each value of s, the number of invertible 2×2 submatrices was counted for any of the resulting invertible random matrices. The invertible matrices, found by the random search, with the maximum number of invertible 2×2 submatrices are provided in [4]. Table 4 shows the largest number of invertible 2×2 submatrices found, in the invertible random matrices.

s	N_2	R_2	0 Frq	1 Frq	2 Frq
3	9	1	$0.\bar{2}$	$0.\bar{4}$	$0.\bar{2}$
4	34	$0.9\bar{4}$	0.25	0.25	0.5
5	86	0.86	0.20	0.48	0.32
6	185	$0.8\bar{2}$	$0.1\bar{6}$	$0.\bar{4}$	$0.3\bar{8}$
7	343	$0.\bar{7}$	≈ 0.204	≈ 0.388	≈ 0.408
8	591	≈ 0.754	≈ 0.156	≈ 0.391	≈ 0.453
9	965	≈ 0.745	≈ 0.173	≈ 0.395	≈ 0.432
10	1479	≈ 0.730	0.18	0.43	0.39
11	2189	≈ 0.724	≈ 0.190	≈ 0.397	≈ 0.413
12	3090	≈ 0.709	≈ 0.188	≈ 0.382	≈ 0.431
13	4306	≈ 0.708	≈ 0.172	≈ 0.402	≈ 0.426

Table 4: The maximum number of invertible 2×2 submatrices and their 2-densities in invertible $s \times s$ random matrices where $s \in \{3, \ldots, 13\}$, for $q = 3$.

3.2 Exhaustive Search and Search for Cyclic and Almost Cyclic Matrices

Similar to the $q = 2$ case, the exhaustive search algorithms and the algorithms for finding cyclic and almost cyclic matrices were used to generate the ternary invertible matrices with the maximum number of invertible 2×2 submatrices. The algorithms are the same as those described in the previous section, only modified to fit the $q = 3$ case. Specifically, the linear independence of the rows of the matrix cannot be checked by means of boolean functions any more. In the exhaustive search, eliminating search branches based on the permutations of columns is limited. In the search for cyclic and almost cyclic matrices on the other hand, in addition to cyclic shifts of each row, multiples of all its cyclic shifts, including itself, were also omitted from the search.

The algorithms were executed on a node on `linux.cs.uwaterloo.ca`, for $3 \leq s \leq 5$ for the exhaustive search and $3 \leq s \leq 20$ for the other two. The exhaustive search for $s = 6$ was computed on 160 cores of a node on RIPPLE server. Tables 5 and 6 present the results.

Table 5 shows, that in most cases, the frequencies of 0's, 1's, and 2's of the cyclic matrices found by the search do not exactly follow the results of the probabilistic analysis. To address this difference, it is required to consider the behavior of function f. Figure 1 demonstrates that the value of function f is mostly sensitive to the changes in the value of γ, rather than changes in the values of α or β. This explains why the frequency of 0's in the cyclic matrices is around 20%, yet the frequencies of 1's and 2's vary considerably.

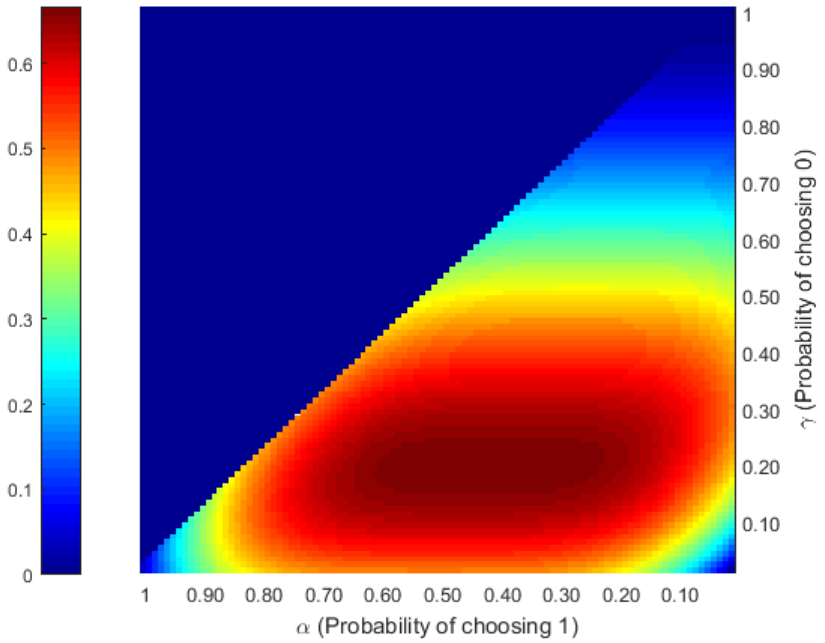
Comparing the results provided in Table 6 to those provided in Table 3, it can be seen that applying the adjusting step improves the results of cyclic search less

often and by smaller values for the $q = 3$ case as it does for the $q = 2$ case. While changing one entry of the binary cyclic matrices can improves the value of N_2 in 10 cases out of first 20 cases and by up to 236, this technique only changes N_2 in only 3 cases out of the first 20 cases in the ternary cyclic matrices. That the ternary matrices have more flexibility than the binary ones and the maximum value of N_2 is naturally higher for them can be explanations for this difference in the performance of this technique on two different sets of matrices.

s	Cyc N_2	Cyc R_2	0 Frq	1 Frq	2 Frq	$N_2(s)$	$R_2(s)$	1 Frq	2 Frq
3	9	1	$0.\overline{3}$	$0.\overline{6}$	0	9	1	$0.\overline{6}$	0
4	34	$0.9\overline{4}$	0.25	0.5	0.25	34	$0.9\overline{4}$	0.625	0.125
5	90	0.9	0.2	0.6	0.2	90	0.9	0.56	0.24
6	189	0.84	$0.1\overline{6}$	0.5	$0.\overline{3}$	195	$0.8\overline{6}$	$0.52\overline{7}$	$0.2\overline{7}$
7	357	≈ 0.810	≈ 0.143	≈ 0.571	≈ 0.286	–	–	–	–
8	600	≈ 0.765	0.25	0.5	0.25	–	–	–	–
9	1008	$0.\overline{7}$	$0.\overline{2}$	$0.\overline{3}$	$0.\overline{4}$	–	–	–	–
10	1550	≈ 0.765	0.2	0.3	0.5	–	–	–	–
11	2288	≈ 0.756	$0.\overline{18}$	$0.4\overline{5}$	$0.\overline{36}$	–	–	–	–
12	3264	≈ 0.749	$0.1\overline{6}$	$0.58\overline{3}$	0.25	–	–	–	–
13	4498	≈ 0.739	≈ 0.154	≈ 0.466	≈ 0.385	–	–	–	–
14	6069	≈ 0.733	≈ 0.214	≈ 0.357	$0\approx 0.429$	–	–	–	–
15	8085	$0.7\overline{3}$	0.2	$0.5\overline{3}$	$0.2\overline{6}$	–	–	–	–
16	10456	≈ 0.726	≈ 0.188	≈ 0.566	0.25	–	–	–	–
17	13413	≈ 0.725	0.177	≈ 0.471	≈ 0.353	–	–	–	–
18	16839	≈ 0.719	$0.1\overline{6}$	$0.3\overline{8}$	$0.\overline{4}$	–	–	–	–
19	20938	≈ 0.716	≈ 0.211	0.421	0.368	–	–	–	–
20	25840	≈ 0.716	0.2	0.5	0.3	–	–	–	–

Table 5: The maximum number of invertible 2×2 submatrices and their 2-densities for $q = 3$ for exhaustive and cyclic search.

Figure 1: The value of function f for different values of α and γ



4 Conclusion

In the previous sections, different methods were presented for generating close to 2-AONT matrices with entries from alphabets $\{0, 1\}$ and $\{0, 1, 2\}$. In this section, we will compare the results of these methods among themselves and with the results from theoretical analysis.

First, we consider the frequencies of 0 and 1 in the binary case, and those of 0, 1, and 2 in the ternary case. For the binary case, it can be observed that the frequency of 1's in the matrices generated by the exhaustive search, or by the search among the cyclic matrices is mostly close to $\frac{1}{\sqrt{2}}$, which is the number that maximizes the expected number of invertible 2×2 submatrices, as calculated by D'Arco et al. [3]. However, for the ternary case, it is only the frequency of 0's that follows the results of calculations for maximizing the expected number of invertible 2×2 submatrices, and stays around $1 - \frac{2}{\sqrt{6}} \approx 0.1835$. On the other hand, it can be noted that the best results found by generating matrices randomly for the ternary case, using the calculated probability distributions for each symbol to occur, achieves R_2 values 0.03 to 0.04 smaller than the corresponding R_2 values of the results among cyclic matrices.

s	Cyc N_2	Cyc R_2	Adj Cyc N_2	Adj Cyc R_2
3	9	1	9	1.0
4	34	$0.9\bar{4}$	34	$0.9\bar{4}$
5	90	0.9	90	0.9
6	189	0.84	189	0.84
7	357	≈ 0.810	357	≈ 0.810
8	600	≈ 0.765	608	≈ 0.776
9	1008	$0.\bar{7}$	1008	$0.\bar{7}$
10	1550	≈ 0.765	1550	≈ 0.765
11	2288	≈ 0.756	2288	≈ 0.756
12	3264	≈ 0.749	3264	≈ 0.749
13	4498	≈ 0.739	4498	≈ 0.739
14	6069	≈ 0.733	6080	≈ 0.734
15	8085	$0.7\bar{3}$	8085	$0.7\bar{3}$
16	10456	≈ 0.726	10482	≈ 0.728
17	13413	≈ 0.725	13413	≈ 0.725
18	16839	≈ 0.719	16839	≈ 0.719
19	20938	≈ 0.716	20938	≈ 0.716
20	25840	≈ 0.716	25840	≈ 0.716

Table 6: Comparison of N_2 and R_2 values for cyclic matrices and matrices with one element adjustment for $s = 3, \ldots, 20$ and the maximum of number of 2×2 submatrices possible in a binary cyclic matrix as an upper bound for $q = 3$.

The next observation is about the behavior of R_2 for the cyclic matrices and $R_2(s)$. In the binary case, the results of the exhaustive search algorithm show that the $R_2(s)$ decreases to about 0.6 as soon as s reaches 9. However, the maximum cyclic R_2 values, that were closely following $R_2(s)$, approach 0.52, which agrees with the result by Zhang et al. [12], that the upper limit of $R_2(s)$ converges to 0.5, as we increase s. This result is also consistent with the expected value of R_2 when we set the frequency of 1 to be $\frac{1}{\sqrt{2}}$ [3] . Although there is not enough data for the ternary case, the results do not rule out the possibility of R_2 converging to $\frac{2}{3}$ when $q = 3$.

Finally, we will discuss the effect of the adjusting step. As mentioned in Section 2, the relative impact of one adjusting step is reduced as s grows. The effect of multiple adjusting steps is open to be studied. Also, the pattern in which it will have no impact, i.e., cyclic matrices are the local maxima in the neighborhood of being 1 entry away, can be further investigated.

Acknowledgements

This work benefitted from the use of the CrySP RIPPLE Facility at the University of Waterloo. Also, N. N. Esfahani thanks Matin Azadmanesh for his helpful comments.

References

[1] R. Canetti, Y. Dodis, S. Halevi, E. Kushilevitz and A. Sahai, Exposure-resilient functions and all-or-nothing transforms, *Adv. in Cryptology, EUROCRYPT 2000*, Springer, (2000), 453–469.

[2] R. G. Cascella, Zh. Cao, M. Gerla, B. Crispo and R. Battiti, Weak data secrecy via obfuscation in network coding based content distribution, *Wireless Days, 2008. WD'08. 1st IFIP*, IEEE, (2008), 1–5.

[3] P. D'Arco, N. Nasr Esfahani and D. R. Stinson, All or nothing at all, *Electron. J. Combin.* **23(4)** (2016), #P4.10.

[4] N. Nasr Esfahani and D. R. Stinson, A list of close to AONT matrices found by computer search, `http://cacr.uwaterloo.ca/techreports/2016/cacr2016-08.pdf`, (2016).

[5] Q. Guo, M. Luo, L. Li and Y. Yang, Secure network coding against wiretapping and Byzantine attacks, *EURASIP J. on Wireless Commun. and Networking*, (2010), 1–5.

[6] A. Proano and L. Lazos. Packet-hiding methods for preventing selective jamming attacks, *IEEE Trans. on Dependable and Secure Comput.* **9** (2010), 101–114.

[7] R. L. Rivest, All-or-nothing encryption and the package transform, In "Fast Software Encryption 1997", E. Biham, ed., *Lect. Notes Comput. Sci.* **1267** (1997), 210–218.

[8] D. R. Stinson, Something about all or nothing (transforms), *Des. Codes Cryptogr.* **22** (2001), 133–138.

[9] R. Vasudevan, A. Abraham and S. Sanyal, A novel scheme for secured data transfer over computer networks. *J.UCS* **11-1** (2005), 104–121.

[10] S. A. Vavasis, *Nonlinear optimization: complexity issues*, Oxford University Press Inc., (1991).

[11] Q. Zhang and L. Lazos, Collusion-resistant query anonymization for location-based services, *IEEE International Conf. Commune.* (2014), 768–774.

[12] Y. Zhang, T. Zhang, X. Wang and G. Ge, Invertible binary matrix with maximum number of 2-by-2 invertible submatrices, *Discrete Math.* **340-2**, (2017), 201–208.

(Received 20 Oct 2016; revised 23 June 2017)