EVALUATION OF GREEDY ALGORITHM OF CONSTRUCTING (0,1)-MATRICES WITH DIFFERENT ROWS[1]

Hasmik Sahakyan, Levon Aslanyan

Abstract: *An approximation greedy algorithm is considered for reconstruction of (0,1)-matrices with different rows. Numbers of pairs of different rows is taken up as a quantitative characteristic, maximization of which, when appropriate, leads to matrices with different rows. Properties of the algorithm are studied and the performance is evaluated based on series of experiments.*

Keywords: *(0,1)-matrices, greedy algorithms.*

ACM Classification Keywords: *F.2.2 Nonnumerical Algorithms and Problems: Computations on discrete structures.*

Introduction

$(0,1)$-matrices with prescribed row and column sums is a classical matter which appears in many branches of applied mathematics. There is a known result by Ryser for a pair of vectors being the row and column sums of a (0,1)-matrix ([R, 1957]). (0,1) matrices with given row and column sums and with special geometric properties/constraints are addressed for example in [DCh, 1999], [BDLNP, 1996], [W, 2001].

Consider the n dimensional unit cube. Vertices of the cube are coded by n-tuples of 0,1 values, and in this way any vertex subset has been presented as a $(0,1)$-matrix, where rows correspond to vertices and thus all rows are different. Row sums indicate the layers of the cube containing the corresponding vertices. i-th column sum identifies the number of vertices in the vertex subset with 1 value in i-th position. Let $R=(r_1,\cdots,r_m)$ and $S=(s_1,\cdots,s_n)$ denote the row and column sum vectors of a $(0,1)$-matrix of size $m\times n$. Now existence of such matrix is equivalent to the existence of m vertices situated in r_1-th, r_2-th, etc. r_m-th layers such that s_1 vertices/tuples contain 1 in the first position, s_2 vertices contain 1 in the second position, etc. and s_n vertices contain 1 in the n-th position. In other words s_i and $m-s_i$ are partition sizes of the vertex subset on i-th direction. In case when the particular positions of vertices are not important and we are just interested in existence of vertex subsets that have given partition sizes – we search out a subclass of matrices with column sums $S=(s_1,\cdots,s_n)$ and with m rows which are different. Both cases (with or without $R=(r_1,\cdots,r_m)$) are known as algorithmically open problems (no polynomial algorithm is known).

For the problem with $S=(s_1,\cdots,s_n)$ and m a greedy algorithm was proposed in [S, 2010 G], which is proven to be optimal in local steps. The current research intends to evaluate performance of this algorithm. Series of experiments made for this purpose, taking into account special properties of the algorithm. We propose that the given greedy algorithm performs a rather good partitioning of matrix columns. For $S=(s_1,\cdots,s_n)$ and m, supposing the existence of row different matrices, greedy algorithm leaves at most 2 length intervals.

$(0,1)$ matrices with different rows

Consider a $(0,1)$-matrix of size $m \times n$. Let $R = (r_1, \cdots, r_m)$ and $S = (s_1, \cdots, s_n)$ denote the row and column sum vectors of the matrix respectively, and let $U(R,S)$ be the class of all $(0,1)$-matrices with row sum R and column sum S. Clearly $0 \le s_i \le m$ for $1 \le i \le n$ and $0 \le r_i \le n$ for $1 \le i \le m$. A necessary and sufficient condition for existence of $(0,1)$ matrices in the class $U(R,S)$ was found by H. J. Ryser. In difference to this case we consider a specific subclass of $U(R,S)$ where all the rows of matrices are different, and in particular, we will consider the class $U(S)$ of all $(0,1)$-matrices with column sum $S = (s_1, \cdots, s_n)$ and with m rows that are all different.

Now we formulate two basic postulations related to the problem posted:

(P1) ***Clarify the Existence issue of*** $(0,1)$ ***matrices in*** $U(S)$,

(P2) ***Construct*** $(0,1)$ ***matrix or matrices in class*** $U(S)$.

We will address, mainly, the construction issues, **(P2)**. Consideration includes also the approximate case and in this context it is reasonable to introduce quantitative characteristics, so that optimized values of such characteristics lead to matrices with different rows in case when the latter exist. As such measure we consider "number of pairs of different rows" and its maximum – which was first considered in [S, 1995]. If $\Im(S)$ denotes the class of $(0,1)$-matrices of size $m \times n$ having column sum vector $S = (s_1, \cdots, s_n)$ (in this way $U(S) \subseteq \Im(S)$), then C_m^2 is the maximum possible number of the pairs of different rows for matrices of $\Im(S)$. This maximum is achieved for matrices of $U(S)$.

Therefore, if $U(S)$ is not empty, then a matrix $M \in \Im(S)$ with maximum number of pairs of different rows will solve the existence problem **(P1)**. In general, when the maximum value is less than C_m^2, then this is an indication that $U(S)$ is empty. Thus, further we will consider the modified postulation:

(P2') **Construct** $(0,1)$ **matrices with** m **rows and the given column sum, that have maximum number of pairs of different rows.**

The requirement of different rows implies a simple restriction on number of possible rows: $m \le 2^n$.

Below we bring some statements/properties from [S, 2009, S, 2010] that we take into account during the experiments.

Lemma 1 [S, 2010]. Suppose that $S = (s_1, \cdots, s_n)$ is the column sum vector for some $m \times n$- matrix with different rows. Then there exists a $(2^n - m) \times n$-matrix with different rows such that $(2^{n-1} - m + s_1, \cdots, 2^{n-1} - m + s_n)$ serves as its column sum vector.

This is a simple but useful property. It says: regardless the issue (existence, construction or characterization of column sum vectors), it is sufficient to consider the case $m \le 2^{n-1}$.

Lemma 2 [S, 2009]. Assume that $U(S)$ is not empty for a given $S=(s_1,\cdots,s_n)$ and suppose that $s_k > m/2$ for some k. Then $U(S')$ is not empty as well for $S'=(s_1,\cdots,s_k-1,\cdots,s_n)$.

Greedy algorithm for solving (P2')

The greedy approach is a recursive execution of a procedure that minimizes/maximizes the increase of the objective function.

Now we describe an algorithm G that constructs a matrix column by column: starting from the first one and adding a column in each step. The objective function is D: $A \in \Im(S) \to$ "number of pairs of differing rows of A"; and the goal is to construct a matrix with the greatest possible value of D, that is $A_{opt} \in \Im(S)$ such that $D(A_{opt}) = \max\limits_{A\in\Im(S)} D(A)$. Let A_G denote a matrix constructed by G and let $A_{G,k}$ is the matrix at the k-th step of G (in this way $A_{G,n} = A_G$). $\Delta D(A_{G,k})$ denote the increase of objective function during the k-th step of G, that is: $\Delta D(A_{G,k}) = D(A_{G,k}) - D(A_{G,k-1})$.

Algorithm *G*

Step1. First column consists of s_1 ones placed in the first s_1 rows-positions followed by $m-s_1$ zeros in others. Two intervals are the result: – the s_1-length interval of ones, and the $(m-s_1)$-length interval of zeros. We denote these intervals by $d_{1,1}^G$ and $d_{1,2}^G$. Hereafter the first sub-index will indicate the number of column and the second – the number of interval within the column. So construction of the first column is in unique way:

$$\begin{cases} d_{1,1}^G + d_{1,2}^G = m \\ d_{1,1}^G = s_1 \end{cases}$$

A pair of rows consists of identical when they belong to the same interval; otherwise it consists of differing rows. Therefore the first column will produce $d_{1,1}^G \cdot d_{1,2}^G$ pairs of different rows. So the increase of objective function is: $\Delta D(A_{G,1}) = d_{1,1}^G \cdot d_{1,2}^G$.

Let we have constructed the first $k-1$ columns. In general, $(k-1)$-th column consists of 2^{k-1} intervals filled by ones and zeros accordingly. Since among them 0-length intervals are possible and they cannot be used anymore, let us assume that the $(k-1)$-th column consists of p non-zero length intervals denoted by $d_{k-1,1}^G, d_{k-1,2}^G, \cdots, d_{k-1,p}^G$. Recall that the rows coincide within the intervals and differ otherwise. If in some column j we get all one length intervals, then at this moment non repetition of all rows, and hence the maximum number of pairs of different rows is already provided. Further constructions can be arbitrary.

Step k. During this step each $d_{k-1,i}^G$ length interval will be partitioned into $d_{k-1,i,0}^G$ and $d_{k-1,i,1}^G$ length intervals filled by zeros and ones respectively: $d_{k-1,i}^G = d_{k-1,i,0}^G + d_{k-1,i,1}^G$ such that

$\sum\limits_{i=1}^{p} d_{k-1,i,0}^G = m - s_k$ and $\sum\limits_{i=1}^{p} d_{k-1,i,1}^G = s_k$.

The increase of objective function during the k-th step is:

$$\Delta D(A_{G,k}) = \sum_{i=1}^{p} d^G_{k-1,i,1} \cdot d^G_{k-1,i,0} .$$

We will release partitions to minimize the length differences of intervals.

The idea is in following: if $s_k = m - s_k, k = 1,\cdots,n$, then in each step we would split every interval into 2 equal (± 1 in case of odd) parts and fill by zeros and ones respectively which will lead to all one length intervals in logarithmic number ([K, 1973]) steps. Furthermore, among all integer partitions of $d^G_{k-1,i}$: $d^G_{k-1,i} = d^G_{k-1,i,1} + d^G_{k-1,i,0}$, the largest product $d^G_{k-1,i,1} \cdot d^G_{k-1,i,0}$ is achieved when $d^G_{k-1,i,1} = d^G_{k-1,i,0}$. Thus following this strategy would bring to the goal, but in general at each step k we have $s_k - (m - s_k)$ extra ones (or $(m - s_k) - s_k$ extra zeros). Trying to be closer to equal lengths of intervals we 1) distribute the extra $s_k - (m - s_k)$ ones (or $(m - s_k) - s_k$ extra zeros) among intervals keeping a "homogeneous" distribution; and then 2) split the remaining intervals into 2 equal parts – putting equal number of zeros and ones.

The lengths of intervals/partitions and their parity will be the same independently of occurrence of extra ones or extra zeros. Thus without loss of generality we may assume that $s_i \geq m - s_i,\ i = 1,\cdots n$.

Now describe the process in detail.

Let $r_k = s_k - (m - s_k)$ and assume that there are l odd length intervals among the intervals of $(k-1)$-th column. It is easy to check that r_k and l have the same parity and hence $r_k - l$ is an even number. Construction of the k-th column is in 2 phases: distribution of r_k "extra" ones during the first, and distribution of remaining ones during the second phases.

1) phase

a) $r_k \leq l$

Chose arbitrary r_k intervals among the l odd intervals and put a 1 in each.

b) $r_k > l$

All l odd intervals get a 1. After this we put ones two by two in intervals starting from the intervals of even length then altering from odd to even, and continuing the process until all r_k ones have been exhausted. If during the process some short intervals have been filled, they do not participate any longer. Let T denote the maximum length of those intervals filled during this process. It is worth to mention that after putting l ones on odd intervals, we get all even lengths, and $r_k - l$ is even as well, so in this way the process of distribution is correct. After this phase there remain equal numbers of zeros and ones.

2) phase

a) $r_k \leq l$

Half of the remaining $l - r_k$ odd intervals get one 0, others – one 1, after that all intervals have been split into equal parts and receive equal number of zeros and ones.

b) $r_k > l$

all intervals have been split into equal parts and receive equal number of zeros and ones.

Let c_i indicates difference between the distributed ones and zeros on i-th interval: $c_i = d^{G_1}_{k-1,i,1} - d^{G_1}_{k-1,i,0},\ i = 1,\cdots,p$.

Thus on k-th column lengths of partitions are the following:

$d^{G_1}_{k-1,i,1} = \frac{d^{G_1}_{k-1,i} + c_i}{2},\quad d^{G_1}_{k-1,i,0} = \frac{d^{G_1}_{k-1,i} - c_i}{2}$ filled by 1 and 0 respectively for $i = 1,\cdots,p$. Each of the intervals may be of 0-length. It is worth to mention that: $\max\limits_{i,j} | c_i - c_j | \le 2$ for all i, j pairs of even $(\ge T)$-length and odd $(\ge T+1)$-length intervals.

Note. The need to choose some quantity of even (odd) intervals among all even (odd) intervals will cause branching during the first phase in case of $r_k > l$. This may bring not uniqueness in constructions.

Nevertheless all branches maximize the increase of the pairs of different rows in each local step.

If the n-th column consists of only 1-length intervals then algorithm G produces a matrix with different rows. Otherwise within intervals of greater lengths – rows are coinciding.

Theorem 1

(1) Each step of the algorithm G is optimal: it provides the maximum increase of the objective function – pairs of differing rows;

(2) All optimal constructions in each column are those according to G.

However algorithm G does not provide the global optimum.

Properties of G

In this section we study properties of G.

For a given vector $S = (s_1,\cdots,s_n)$ consider the class $U(S)$ of all $(0,1)$-matrices with column sum $S = (s_1,\cdots,s_n)$ and with m rows which are all different. Suppose that $U(S)$ is not empty and $A \in U(S)$. Interchanging ones with zeros in some column i of A will not produce repeating rows and hence it will lead to a matrix A^i that belongs to the class $U(S^i)$ where $S^i = (s_1,\cdots,m - s_i,\cdots,s_n)$. So A is easily transformable to A^i and vice versa. Thus having a goal the existence/construction issues in $U(S)$ we assume without loss of generality that all $s_i \ge m/2$. Concerning the order of components of S, - it is not important while considering the existence problem. As regards the construction process, it may have an influence. However we will study

properties of the algorithm assuming that $s_1 \geq \cdots \geq s_n$. Thus hereafter in this paper we will assume that $s_1 \geq \cdots \geq s_n$ and $s_i \geq m/2$ for $i = 1,\cdots,n$.

Below we will prove a property of algorithm G consisting in following:

Theorem 2. If for a vector $S = (s_1,\cdots,s_n)$ algorithm G constructs a matrix from $U(S)$, and $s_k > m/2$ for some k, then G constructs also a matrix from $U(S')$, where $S' = (s_1,\cdots,s_k - 1,\cdots,s_n)$ [1].

While Lemma 2 states that "good" vectors supporting existence of matrices in $U(S)$ are those having coordinate values close to $m/2$, the theorem 2 confirms that these are "good" also from the point of view of construction by G.

Before proving the theorem we bring some notes/properties concerning the constructions of algorithm G taking into account the order $s_1 \geq \cdots \geq s_n$:

Quantities of intervals increase from column to column - it is obvious.

Quantities of odd intervals do not decrease from column to column - indeed, each odd interval being split produces one odd interval and each even interval produces 0 or 2 odd intervals.

If during the construction of some column an interval (say length d) receives some number of extra 1s (say c) then all partitions of d in the consequent columns receive $\leq c$ extra 1s (independent of size of partition) - this follows from the constructions.

Now let us prove Theorem 2.

G constructs a matrix from $U(S)$ means that on the n-th column it receives all 1-length intervals. Now we prove that starting with $S' = (s_1,\cdots,s_k - 1,\cdots,s_n)$, G will construct all 1-length intervals as well.

The first $k-1$ columns constructed by G are the same for S and S'. Now consider constructions of the k-th columns by s_k and $s_k - 1$ respectively (denoted them by $G(s_k)$ and $G(s_k - 1)$) and convince that in the latter case it will bring to a matrix from $U(S')$. It is worth to mention that $s_k > s_{k+1}$.

Assume that there are l odd length intervals in $(k-1)$-th column and r_k is the difference between ones and zeros for the k-th column: $r_k = s_k - (m - s_k)$. r_k and l have the same parity and hence $r_k - l$ is even number. k-th column had been constructed in 2 phases: distribution of r_k "extra" ones during the first, and distribution of remaining ones during the second phases.

Now let r'_k denote the new difference: $r'_k = (s_k - 1) - (m - (s_k - 1)) = s_k - (m - s_k) - 2$ so $r'_k = r_k - 2$.

---

[1] *If equal components exist in S then choosing $s_k > m/2$ to decrease it by 1, we take the last such coordinate keeping the vector $S' = (s_1,\cdots,s_k - 1,\cdots,s_n)$ ordered decreasingly.*

We will follow constructions of k-th column for both r_k and $r_k - 2$ extra ones, and compare the lengths of intervals.

Consider possible cases:

a) $r_k \le l$ hence $r'_k \le l$

Constructions and reasoning are the same as for s_k:

Phase 1

Chose arbitrary r'_k intervals among the l odd intervals and put a 1 in each.

Phase 2

Half of the remaining $l - r'_k$ odd intervals get one 0, others get one 1, after that all intervals have been split into equal parts and receive equal number of zeros and ones.

Lengths had not been changed and hence constructions of remaining columns will be the same which leads to a matrix from $U(S')$.

b) $r_k > l$

Since $r_k - l$ is even number, then $r_k - l \ge 2$. Consider cases:

1. $r'_k \le l$

It follows that $r'_k = l$. This means that allocating 1 to each odd interval, all r'_k extra ones are exhausted in $G(s_k - 1)$:

Phase 1

All l odd intervals get a 1 value.

Phase 2

All intervals have been split into equal parts and receive equal number of zeros and ones.

Meanwhile in $G(s_k)$ two extra 1s remained for distributing among even intervals.

We denote by d the length of an even interval which received two extra 1s in $G(s_k)$. Since other lengths are the same in both $G(s_k - 1)$ and $G(s_k)$, we will follow only partitions of d.

$$G(s_k): \quad d \;\begin{cases} (d+2)/2 = d/2 + 1 \\ (d-2)/2 = d/2 - 1 \end{cases}$$

$$G(s_k - 1): d \;\begin{cases} d/2 \\ d/2 \end{cases} \qquad (1)$$

Consider cases:

- $d/2$ **is odd**

Then $d/2+1$ and $d/2-1$ are even.

Since the order of coordinates in S is decreasing and $s_k > s_{k+1}$, then in both cases these intervals do not receive extra ones. In $(k+1)$-th column we get the following partitions coming from d :

$G(s_k)$: $(d/2+1)/2$, $(d/2+1)/2$, $(d/2-1)/2$, $(d/2-1)/2$, and

$G(s_k-1)$: $(d/2+1)/2$, $(d/2-1)/2$, $(d/2+1)/2$, $(d/2-1)/2$.

So – they bring to the same lengths.

- $d/2$ **is even**

$d/2+1$ and $d/2-1$ are odd.

In $(k+1)$-th column we will get:

$G(s_k)$: $(d/2+1+1)/2$, $(d/2+1-1)/2$, $(d/2-1+1)/2$, $(d/2-1-1)/2$, and

$G(s_k-1)$: $d/4$, $d/4$, $d/4$, $d/4$. Thus:

$$G(s_k): \; d/2 \;\begin{cases} d/4+1 \\ d/4-1 \end{cases}$$

$$G(s_k-1): \; d/2 \;\begin{cases} d/4 \\ d/4 \end{cases} \qquad (2)$$

Depending on parity of $d/4$, the process will be continued altering within the above cases. It stops in some column $(k+t)$ where $d/(2^t)$ is the first odd length and in this way the interval lengths for $G(s_k)$ and $G(s_k-1)$ are the same. Otherwise $d=2^q$ for some q and process stops in $(k+q)$ column where all lengths are 1. It will happen before the column n, since by assumption G constructs a matrix from $U(S)$.

2. $r'_k > l$

Phase 1

All l odd intervals get a 1. After this we put ones two by two in intervals starting from the intervals of even length then altering from odd to even, and continuing cyclically until all r_k ones have been exhausted.

Phase 2

all intervals have been split into equal parts and receive equal number of zeros and ones.

Suppose that phase 1 stopped on odd intervals:

$G(s_k)$: t of odd intervals received c extra ones (c is odd), others received $c-2$ or filled completely, even lengths either received $c-1$ or are filled;

$G(s_k-1)$: $t-1$ odd intervals received c extra ones, others received $c-2$ extra ones or filled completely, even lengths either received $c-1$ or filled.

We denote by d the length of an odd interval which received extra two 1s in $G(s_k)$. Since other lengths are the same in both $G(s_k-1)$ and $G(s_k)$, we will follow only partitions of d.

$$G(s_k): \quad d \;\begin{cases} (d+c)/2 \\ (d-c)/2 \end{cases}$$

$$G(s_k-1): d \;\begin{cases} (d+c-2)/2 = (d+c)/2-1 \\ (d-(c-2))/2 = (d-c)/2+1 \end{cases}$$

Phase 1 stopped on even intervals:

$G(s_k)$: q of even intervals received c (c is even) extra ones, others received $c-2$ or filled completely, odd lengths either received $c-1$ or filled;

$G(s_k-1)$: $q-1$ even intervals received c extra ones, others received $c-2$ extra ones or filled completely and odd lengths received $c-1$ or filled.

We denote by d the length of an odd interval which received the extra two 1s in $G(s_k)$. Other lengths are the same. The picture is as in previous case.

For easy construction we bring the details of the following particular case. In general, the reasoning is in an analogous way.

Suppose that r'_k is just the number of odd intervals + 2 times the number of even intervals.

Phase 1

All odd intervals get a 1 and all even intervals get two 1s.

Phase 2

All intervals have been split into equal parts and receive equal number of zeros and ones.

Then one interval receives 3 extra 1s in $G(s_k)$. We denote by d the length of this interval. Again other lengths are the same and we will follow only partitions of d.

$$G(s_k): \quad d \;\begin{cases} (d+3)/2 = (d+1)/2+1 \\ (d-3)/2 = (d-1)/2-1 \end{cases}$$

$$G(s_k-1): d \;\begin{cases} (d+1)/2 \\ (d-1)/2 \end{cases}$$

Consider cases:

$(d+1)/2$ is odd and $(d-1)/2$ is even.

Then $(d+1)/2+1$ is even and $(d-1)/2-1$ is odd.

Since $s_k > s_{k+1}$, then the number of extra ones decreased by two and taking into account that the number of odd intervals did not decrease, we make the following conclusion: in both cases all odd intervals get at most one extra 1 in both cases. Even intervals will get either 2 or 0 extra ones (we will assume that the same intervals in both cases get the same amount of extra ones).

First consider case of 0 extra ones on even intervals. So in $(k+1)$-th column we get the following partitions coming from d :

$G(s_k)$: $((d+1)/2+1)/2$, $((d+1)/2+1)/2$, $((d-1)/2-1+1)/2$, $((d-1)/2-1-1)/2$

$G(s_k-1)$: $((d+1)/2+1)/2$, $((d+1)/2-1)/2$, $(d-1)/4$, $(d-1)/4$.

So the lengths are:

$G(s_k)$: $(d-1)/4+1$, $(d-1)/4+1$, $(d-1)/4$, $(d-1)/4-1$

$G(s_k-1)$: $(d-1)/4+1$, $(d-1)/4$, $(d-1)/4$, $(d-1)/4$.

Thus all intervals of $(k+1)$-th column are the same in both $G(s_k)$ and $G(s_k-1)$ except the following lengths: $(d-1)/4+1$ and $(d-1)/4-1$ in $G(s_k)$; $(d-1)/4$ and $(d-1)/4$ in $G(s_k-1)$.

The continuations are as in (2).

Now consider the case of 2 extra ones on even intervals. Partitions in $(k+1)$-th column coming from d are the following:

$G(s_k)$: $((d+1)/2+1+2)/2$, $((d+1)/2+1-2)/2$, $((d-1)/2-1+1)/2$, $((d-1)/2-1-1)/2$

$G(s_k-1)$: $((d+1)/2+1)/2$, $((d+1)/2-1)/2$, $((d-1)/2+2)/2$, $((d-1)/2-2)/2$

So all intervals in $(k+1)$-th column are of the same length except the following:

$G(s_k)$: $(d-1)/4+2$ and $(d-1)/4$

$G(s_k-1)$: $(d-1)/4+1$ and $(d-1)/4+1$.

Continuations are in analogous way.

$(d+1)/2$ is even and $(d-1)/2$ is odd.

Then $(d+1)/2+1$ is odd and $(d-1)/2-1$ is even.

This is similar to previous cases.

Evaluation of Results – Experiments

Before experimental treatment of algorithm G we refer to a special subset of $U(S)$, - class of matrices with column sums obeying the property of being a "boundary vector".

Definition [S, 2009]. Let ψ_m denote the set of all vectors $S = (s_1, \cdots, s_n)$ with $0 \le s_i \le m$, such that $U(S)$ is not empty. Vector $\widehat{S} \in \psi_m$ is called an (upper) boundary vector for ψ_m if no vector greater than $\widehat{S}$ belongs to ψ_m.

Further the boundary vectors are found among the vectors that correspond to monotone Boolean functions.

It follows from the Theorem 2 that more "difficult" vectors for the algorithm G are boundary ones. Thus for estimation of the performance of G it is reasonable to address this set of vectors.

All monotone Boolean functions are composed for $n = 6$ with $m = 1, \cdots, 2^{n-1}$ one values. For example for $m = 28$ there are 390050 functions. Corresponding column sum vectors are calculated. As different matrices may have the same column sum vector, there were repetitions among them. Besides, only vectors with decreasing order of coordinates are taken. Because algorithm G is sensitive to this order, it may depreciate the ability of algorithm. Further in case of branching in some column, among all possible partitions only one is selected (an arbitrary choice), that also may decrease abilities of the algorithm. However all these steps simplify the calculations.

Algorithm G succeeds if in the last column of constructions only one length intervals occur. Experiments show that for some cases in the last column there remain 2-length intervals, in most cases only one such interval is present. Table 1 shows the result.

Table 1.

Number of rows	Number of vectors	One 2-length interval	More 2 length intervals	Number of rows	Number of vectors	One 2-length interval	More 2 length intervals
<12		0	0	22	172	59	1
12	23	5	0	23	193	63	3
13	30	5	0	24	214	90	2
14	41	11	1	25	232	99	3
15	51	11	0	26	255	119	2
16	69	18	0	27	265	108	5
17	83	26	1	28	290	134	5
18	97	33	1	29	287	119	11
19	115	34	0	30	287	159	3
20	134	40	1	31	284	121	15
21	149	44	1	32	253	0	0

Besides the case $n = 6$, special "bad" vectors are composed from the point of view of the algorithm G where:

- decrease order is the worst for the given vector
- there is only one successful choice of partitioning in case of branching, and we didn't take it, etc.

On the other hand series of easy constructible monotone Boolean functions are considered in up to 11 dimensional unit cubes and column sums are calculated for corresponding $(0,1)$-matrices. The result of performance of algorithm G for all these vectors remained unchanged: there are no intervals of > 2 length in the last column.

At this moment it is only a supposition based on experiments that algorithm G will not produce matrices with > 2-length intervals in the last column. Further efforts are required to prove or reject this supposition. However assuming true, this property let us estimate the performance of G.

The worst case with this assumption is a matrix with all 2-length intervals in the last column. This implies $m/2$ pairs of coinciding/repeating rows in the matrix. Denote by m_G the number of different rows by G and taking into account that there are C_m^2 different pairs of rows in matrices with m different rows, we get $m_G = m(m-1)/2 - m/2 = m/2(m-2)$. Let A_{opt} be an optimal algorithm constructing target matrices and hence $m_{A_{opt}} = m(m-1)/2$. Compose the performance ratio: $\frac{m_{A_{opt}}}{m_G} = 1 + \frac{1}{m-2}$.

Bibliography

[DCh, 1999] Durr Ch., Chrobak M., Reconstructing hv-convex polyominoes from orthogonal projections, Information Processing Letters 69 (1999) pp. 283-291.

[BDLNP, 1996] E. Barcucci, A. Del Lungo, M. Nivat, and R. Pinzani, Reconstructing convex polyominoes from horizontal and vertical projections, Theoret. Comput. Sci., 155:321{347, 1996.

[W, 2001] G.J. Woeginger, The reconstruction of polyominoes from their orthogonal projections, Inform. Process. Lett., 77:225{229, 2001.

[R, 1966] H. J. Ryser, Combinatorial Mathematics, 1966.

[K, 1973] D. Knuth, The Art of Computer Programming, vol.3. Sorting and Searching, Addison-Wesley Publishing Company, 1973.

[S, 1995] H. Sahakyan, Hierarchical Procedures with the Additional Constraints, II Russia, with participation of NIS Conference, Pattern Recognition and Image Analysis: New Information Technologies, Ulianovsk, 1995, pp.76-78.

[S, 2009] H. Sahakyan, "Numerical characterization of n-cube subset partitioning", Discrete Applied Mathematics, vol. 157, pp. 2191-2197, 2009.

[S, 2010] H. Sahakyan, "Boundary elements of the n-cube subset partitioning characterization", submitted to Discrete Applied Mathematics.

[S, 2010 G] H. Sahakyan, "Approximation greedy algorithm for reconstruction of (0,1)-matrices with different rows"

Authors' Information

Hasmik Sahakyan – Leading Researcher, Institute for Informatics and Automation Problems, NAS RA, P.Sevak St. 1, Yerevan 14, Armenia, e-mail: hasmik@ipia.sci.am

Levon Aslanyan – Head of Department, Institute for Informatics and Automation Problems, NAS RA, P.Sevak St. 1, Yerevan 14, Armenia, e-mail: lasl@.sci.am