Serdica J. Computing **2** (2008), 349–368

Serdica
Journal of Computing

Bulgarian Academy of Sciences
Institute of Mathematics and Informatics

CONSTRUCTING A CANONICAL FORM OF A MATRIX IN SEVERAL PROBLEMS ABOUT COMBINATORIAL DESIGNS*

Zlatka Teneva Mateva

Abstract. The author developed computer programs needed for the classification of designs with certain automorphisms by the local approach method. All these programs use canonicity test or/and construction of canonical form of an integer matrix. Their efficiency substantially influences the speed of the whole computation. The present paper deals with the implemented canonicity algorithm. It is based on ideas used by McKay, Meringer, Kaski and Bouyukliev, but while their algorithms are for the equivalence test, the canonicity test or finding canonical representative of only one type of combinatorial object (graph, code, design, binary matrix, etc.), the algorithm presented in this paper is meant to work fast on all types of integer matrices used for the classification of designs with predefined automorphisms. This is achieved through the suitable *spectrum* invariant, and the way it is used to cut off some branches of the search tree.

ACM Computing Classification System (1998): F.2.1, G.2.2.
Key words: Algorithm, automorphism, incidence matrix, orbit matrix, group action, canonical form, BIBD.

*Partially supported by the Bulgarian Science Fund contract with TU Varna, No 487.

1. Introduction.

1.1. Construction and study of incidence structures. Incidence structures and the integer matrices associated with them are the subject of a dynamically developed field of modern Combinatorics. Usually an equivalence relation is defined over the set of such combinatorial objects and the corresponding equivalence classes are considered. The main problems are:

▶ **Existence problem** – find one such object, or prove its nonexistence.

▶ **Exhaustive search** – find at least one element of each equivalence class.

▶ **Filtering away equivalent objects** – leave exactly one element of each equivalence class. Sometimes only one element has some additional properties and is called the canonical representative of this class.

▶ **Determining the automorphism group** – the group of all equivalence transformations of the object.

In most cases the solution of open problems is only possible with the help of computers and this makes the problem of finding effective and correct algorithms very important.

1.2. About the different problems that are the subject of this paper. All the problems discussed here can be solved using one and the same *main algorithm* for finding the canonical form of a matrix under a definite group of equivalence transformations.

The problem which this *main algorithm* solves is a generalization of the problem of finding the canonical form of an $(m \times n)$ matrix of a certain type under the action of the permutation group $S_m \times S_n$ considered, for instance, in [24], [11], [15], [13], [14], [4], and is close to the canonicity test in [26], [6], [7], [25].

The main difficulty in solving such problems arises from the great number of permutations which have to be considered. This is why the authors of the most popular software products as DISKRETA [8], MOLGEN [9], QEXTENTION [3], MAGMA [2], and the subsystems DESIGN [28] and GRAPH [27] in GAP apply many of techniques to reduce this number. The *main algorithm* presented here implies ideas used by Brendan McKay [22] in the isomorphism test of graphs, by Marcus Meringer [15] in the canonicity test of molecular graphs, by Petery Kaski [12] in the canonicity test and canonical representative construction for designs and design resolutions and by Iliya Bouyukliev [4] in the isomorphism test of codes and designs, as well as in the equivalence test for Hadamard matrices.

1.3. Computer programs for the classification of designs with certain automorphisms. Since the construction and study of combinatorial designs is presently done by the help of computers, it is very important that at least part of the results should be obtained in several ways by different software, and ideally by different algorithms. For that purpose the author developed computer programs needed for the classification of designs with certain automorphisms by the local approach method, i.e, these are programs for construction of orbit matrices and their extension to combinatorial designs, as well as for the determination of the automorphism group and other design properties. They were used in parallel with Topalova's programs (described in [29]) for obtaining and partially verifying the results in [16], [18], [19], [17], and [20].

In the author's approach the canonicity test or the construction of canonical form of an integer matrix is a very important part of all these programs and substantially influences the speed of the whole computation. The specifics of its implementation for all these design construction problems are the subject of the present paper.

2. Necessary definitions and theorems.

Sets, partitions, groups

2.1. Sets. Let $\mathbb{N}_p = \{0, 1, 2, \ldots, p\}$ and $\mathbb{N}_p^+ = \{1, 2, \ldots, p\}$. Let $\mathbb{M}_{m,n}(\mathbb{N}_p)$ be the set of all $(m \times n)$ matrices with elements of $\mathbb{N}_p$. Let us define a lexicographic order in $\mathbb{M}_{m,n}(\mathbb{N}_p)$.

Let $A = (a_{i,j})$ be an arbitrary matrix of the set $\mathbb{M}_{m,n}(\mathbb{N}_p)$. Denote by a'_i and a''_j, $i \in \mathbb{N}_m^+, j \in \mathbb{N}_n^+$ respectively the i-th row and j-th column of the matrix A, and by $A(i_1, \ldots, i_s) \in \mathbb{M}_{s,n}(\mathbb{N}_p)$ the part of A formed by the rows $a'_{i_1}, \ldots, a'_{i_s}$ in this order. Note that the row (column) vectors of rows (columns) a'_{n_1} and a'_{n_2} (a''_{n_1} and a''_{n_2}) might be the same, yet we consider the two rows (columns) different, i.e, we always associate with them the row (column) numbers n_1 and n_2. On the other hand, we define on the set of rows (columns) a partial order relation '$<$' such that $a'_{n_1} < a'_{n_2}$ ($a''_{n_1} < a''_{n_2}$) iff the row (column) vector of row (column) a'_{n_1} (a''_{n_1}) is lexicographically smaller than that of a'_{n_2} (a''_{n_2}). For simplicity we will often refer to the row (column) number instead of the row (column) itself, i.e, we will talk of row (column) n instead if a'_n (a''_n).

2.2. Partitions. In combinatorial mathematics, an *ordered partition* π of a set Ω is a sequence $\{\Omega_1, \ldots, \Omega_s\}$ of disjoint nonempty subsets of Ω whose union is $\cup_{i=1}^{s} \Omega_i = \Omega$. This differs from partition of a set in that the order of the Ω_i matters. An ordered partition π is *regular*, if for any two different cells Ω_i and Ω_j, $i < j$ iff $\min\{\Omega_i\} < \min\{\Omega_j\}$.

When the elements of Ω have been assigned the numbers from 1 to $|\Omega|$, each partition π of Ω can be considered as a partition of the set $\mathbb{N}^+_{|\Omega|}$ and respectively each partition of the set $\mathbb{N}^+_{|\Omega|}$ defines a partition of the set Ω. The elements of the partition are called *cells*.

Consider a partition of the set of rows (columns) of a matrix. A cell of which all elements have one and the same row (column) vector is called *simple*, and a partition of which all cells are simpl, is called *discrete*.

A one-to one correspondence can be defined between each partition $\pi = \{\Omega_1, \dots, \Omega_s\}$ and a surjective function $\widetilde{\pi} : \Omega \longrightarrow \pi$, defining a correspondence of each element $\omega_i \in \Omega$ to the cell $\Omega_j \in \pi$ containing it.

A permutation $\varphi \in S_{|\Omega|}$ of the elements of Ω does not change the partition $\pi = \{\Omega_1, \dots, \Omega_s\}$, if for each element $\omega \in \Omega$ the element $\varphi\omega$ is from the cell Ω_i iff $\omega \in \Omega_i$. All permutations of the integers from 1 to m which do not change the partition π of the set $\mathbb{N}^+_m$ form a group denoted by $S_m(\pi)$.

If π and η are two partitions of the set Ω, we say that η is *finer* than π or that π is *coarser* than η, and we write $\eta \preccurlyeq \pi$, if each element of η is a subset of an element of π.

2.3. Permutation groups acting on $\mathbb{M}_{m,n}(\mathbb{N}_p)$. The subgroup G' of the symmetric group of order m ($G' \leq S_m$) *acts on the rows* of the matrices of the set $\mathbb{M}_{m,n}(\mathbb{N}_p)$, if a correspondence is defined between each ordered pair (ρ, A), where $\rho = (\rho_1, \dots, \rho_m) \in G'$, $A = (a'_1, \dots, a'_m)^T \in \mathbb{M}_{m,n}(\mathbb{N}_p)$ and the matrix $\rho A = (a'_{\rho_1}, \dots, a'_{\rho_m})^T$. The group $G'' \leq S_n$ *acts on the columns* of the matrices of the set $\mathbb{M}_{m,n}(\mathbb{N}_p)$, if $\forall \sigma = (\sigma_1, \dots, \sigma_n) \in G''$ and $\forall A = (a''_1, \dots, a''_n) \in \mathbb{M}_{m,n}(\mathbb{N}_p)$ it holds $\sigma A = (a''_{\sigma_1}, \dots, a''_{\sigma_n})$. The group $G' \times G''$ *acts on the rows and columns* of the matrices of the set $\mathbb{M}_{m,n}(\mathbb{N}_p)$ juxtaposing to each matrix $A \in \mathbb{M}_{m,n}(\mathbb{N}_p)$ and to each pair of permutations $(\rho, \sigma) \in G' \times G''$ the matrix $(\rho, \sigma)A = \rho(\sigma A) = \sigma(\rho A)$, i.e, the matrix obtained from A by applying the permutation ρ on the rows, and σ on the columns.

The elements of G' are called *row permutations*, of G'' – *column permutations*, and of $G' \times G''$ – *matrix permutations.*

Let the groups G' and G'' act respectively on the rows, and columns of $\mathbb{M}_{m,n}(\mathbb{N}_p)$. The two matrices A and B of the set $\mathbb{M}_{m,n}(\mathbb{N}_p)$ are *equivalent under the group* $G = G' \times G''$ ($A \overset{G}{\sim} B$) if a matrix permutation $(\rho, \sigma) \in G$ exists, such that

$$A = (\rho, \sigma)B, \tag{1}$$

i.e, when the matrices A and B are of one and the same orbit of $\mathbb{M}_{m,n}(\mathbb{N}_p)$ under the action of G.

Matrix permutations for which (1) holds are *isomorphic* ones, transforming the matrix B into A. An isomorphism transforming A into itself is an *automorphism* of A, and the set of all automorphisms of A under G form a group which is called $Aut(A, G)$.

If a matrix permutation (ρ, σ) is an automorphism, then if you know ρ, it is trivial to determine σ up to a permutation of columns with equal column vectors in each cell. That is why below we often find automorphism row permutations, and do not explain anything about the related column permutations.

3. Construction of the canonical form of a matrix under a certain group of permutations.

The problem

3.1. Let the regular partitions π' and π'' of the sets $\mathbb{N}_m^+$ and $\mathbb{N}_n^+$, define partitions respectively of the sets of the rows, and of the columns of an arbitrary matrix A of $\mathbb{M}_{m,n}(\mathbb{N}_p)$.

The problem is, for given partitions π' and π'' for an arbitrary matrix $A \in \mathbb{M}_{m,n}(\mathbb{N}_p)$, to find the matrix

$$c(A) = \max\{\tau A \mid \tau \in G\}, \tag{2}$$

where G is the group of all matrix permutations which do not change both partitions.

The matrix $c(A)$ is lexicographically the maximal matrix of the orbit of A under the action of the group $G = S_m(\pi') \times S_n(\pi'')$ over the set $\mathbb{M}_{m,n}(\mathbb{N}_p)$.

3.2. Proposition. *The correspondence $c : \mathbb{M}_{m,n}(\mathbb{N}_p) \longrightarrow \mathbb{M}_{m,n}(\mathbb{N}_p)$, defined by the equality* (2) *is canonical.*

Proof. The mapping c is well defined, because for any matrix $A \in \mathbb{M}_{m,n}(\mathbb{N}_p)$ the finite set $\{\tau A \mid \forall \tau = (\rho, \sigma) \in G\}$ contains its exact upper bound $c(A)$ and answers the canonicity requirements:

$c(A) \overset{G}{\sim} A$ and

$c(A) = c(B)$ iff $A \overset{G}{\sim} B$,

which shows that the correspondence c defined by the equality (2) is a canonical mapping. $\square$

The image $c(A)$ of the matrix $A \in \mathbb{M}_{m,n}(\mathbb{N}_p)$ by the canonical mapping c is the canonical representative of the orbit of the matrix A under the action of the group $G = S_m(\pi') \times S_n(\pi'')$ over the set $\mathbb{M}_{m,n}(\mathbb{N}_p)$.

Definition. *The matrix $c(A)$, defined by the equality* (2) *is the canonical form of the matrix A under the group G.*

The partitions π' and π'' allow us to add new restrictions for the group G, which makes the algorithm more applicable.

Without loss of generality we can work with regular partitions.

3.3. The **trivial algorithm** solving problem 3.1 is contained in its definition, and implies applying of all matrix permutations of the group $G = S_m(\pi') \times S_n(\pi'')$ on A and choosing the maximal of the obtained matrices. To avoid saving of all the matrices τA , $\tau \in G$ only the currently maximal matrix $\widetilde{A}$ is saved. Unfortunately, this approach is only applicable for very small groups of permutations.

Mathematical basis of the main algorithm

3.4. Let π'' be a partition of the column set of the matrices of $\mathbb{M}_{m,n}(\mathbb{N}_p)$. Of special importance for the algorithm is the function $s(A, \pi'') : \mathbb{M}_{m,n}(\mathbb{N}_p) \to \mathbb{M}_{m,n}(\mathbb{N}_p)$, which transforms a matrix $A \in \mathbb{M}_{m,n}(\mathbb{N}_p)$ into the matrix that is obtained from A by sorting of the columns in the cells of the partition π'' in descending order.

Proposition. *If π'' and η'' are two partitions of the set $\mathbb{A}''$ of the columns of a matrix $A \in \mathbb{M}_{m,n}(\mathbb{N}_p)$ and η'' is finer than π'', then the matrix $sort(A, \pi'')$ is greater or equal to the matrix $s(A, \eta'')$, i.e. if $\eta'' \preccurlyeq \pi''$ then $s(A, \eta'') \leqslant s(A, \pi'')$.*

The proof of this proposition follows directly from the way the matrices $s(A, \eta'')$ and $sort(A, \pi'')$ are obtained.

3.5. The next proposition is used in almost all similar algorithms.

Proposition. *The canonical matrix $c(A)$ is equal to the greatest matrix in the set*

$$\{s(\rho A, \pi'') \mid \rho \in S_m(\pi')\}.$$

Proof. Let $\tau = (\rho, \sigma)$ be a pair of permutations which does not change the partitions π' and π''. Then

$$\tau A \leq s(\tau A, \pi'') = s((\rho, \sigma)A, \pi'') = s(\rho(\sigma A), \pi'') = s(\rho A, \pi''),$$

Therefore

$$c(A) = \max\{\tau A | \tau \in G\} = \max\{s(\rho A, \pi'') | \rho \in S_m(\pi'')\}. \qquad \square$$

Corollary. *To find the matrix $c(A)$ it is enough to consider only the elements of the group $S_m(\pi')$. For each $\rho \in S_m(\pi')$ the matrix $s(\rho A, \pi'')$ is determined by some fast sorting algorithm.*

Applying this consequence makes the number of the tested permutations $|S_n(\pi'')|$ times smaller.

3.6. The next proposition allows us to drop some of the permutations of the group $S_m(\pi')$. It follows directly from the lexicographic order in $\mathbb{M}_{m,n}(\mathbb{N}_p)$.

Proposition. *If A and B are two matrices of the set $\mathbb{M}_{m,n}(\mathbb{N}_p)$ and $A < B$, then for each natural number $s \in \mathbb{N}_m^+$ it holds*

$$A(1,2,\dots,s) \le B(1,2,\dots,s).$$

Corollary. $c(A)(1,\dots,t) \ge \widetilde{A}(1,\dots,t)$ *for each $s \in \mathbb{N}_m^+$.*

Corollary. *If for the permutation $\rho = (\rho_1,\dots,\rho_t,\dots,\rho_m) \in S_m(\pi')$ a natural number $t \in \mathbb{N}_m^+$ exists, such that $s(\rho\widetilde{A}(1,\dots,t),\pi'') < \widetilde{A}(1,\dots,t)$, then for each permutation $\varphi = (\rho_1,\dots,\rho_t,\varphi_{t+1},\dots,\varphi_m)$ it holds that $s(\varphi\widetilde{A},\pi'') < \widetilde{A}$.*

All permutations φ from Consequence 2, can be omitted because $\varphi\widetilde{A} < \widetilde{A} \le c(A)$ for them. The smaller the number t of the position, in which the difference is found, the bigger the number of omitted permutations. Their number is greatest when the canonical matrix $c(A)$ is obtained by the permutation considered first. Note that even then the algorithm cannot finish, because it is not proved that this is the matrix wanted. All permutations for which $c(A) = s(\rho A, \pi'')$ have to be checked. i.e, all permutations of the automorphism group of the canonical matrix $c(A)$.

3.7. The next proposition by Grund [15] allows us to cut off many branches of the search tree of the group $S_m(\pi')$, each time when a new automorphism of the matrix $\widetilde{A}$ is found.

Proposition. *Let the group G operate on the nonempty set Ω and let $\Gamma \le G$. Consider $\omega \in \Omega$ such that $\varphi\omega \le \omega$ for each $\varphi \in \Gamma$ and let there exist $\widehat{\varphi} \in \Gamma$ and $\psi \in G$ for which $\widehat{\varphi}\psi\omega = \omega$. Then*

$$\varphi\psi\omega \le \omega, \quad \forall \varphi \in \Gamma. \tag{3}$$

Proof. Let $\varphi \in \Gamma$. Then $\varphi\psi\omega = \varphi\widehat{\varphi}^{-1}\widehat{\varphi}\psi\omega = \varphi\widehat{\varphi}^{-1}\omega \le \omega$, because $\varphi\widehat{\varphi}^{-1} \in \Gamma$. □

Corollary. *Let $\widetilde{A}$ be a matrix of $\mathbb{M}_{m,n}(\mathbb{N}_p)$. Let Γ be a subgroup of the group $S_m(\pi')$ and let $s(\varphi\widetilde{A}, \pi'') \le \widetilde{A}$ for each $\varphi \in \Gamma$. If $\psi \in S_m(\pi')$ and $\widehat{\varphi} \in \Gamma$ exist, such that $\widehat{\varphi}\psi\widetilde{A} = \widetilde{A}$, then for each $\varphi \in \Gamma$ it holds $s(\varphi\psi\widetilde{A}, \pi'') \le \widetilde{A}$.*

Proof. Follows from the last proposition by setting $G = S_m(\pi')$, $\Omega = \mathbb{M}_{m,n}(\mathbb{N}_p)$ and $\omega = \widetilde{A}$. $\square$

Namely, when all elements $\varphi \in \Gamma$ of the group $\Gamma \le S_m(\pi')$ have been checked and no counter-example to the maximality of the matrix $\widetilde{A}$ has been found, then finding a permutation of the rows $\psi \in S_m(\pi') \backslash \Gamma$, which is an automorphism, allows further skipping of the elements of the right coset $\Gamma\varphi$.

3.8. Proposition. *Let $\mathcal{A} < S_m(\pi')$ be a group of automorphism row permutations of the matrix A, $B = s(\rho A, \pi'')$ for some $\rho \in S_m(\pi') \setminus \mathcal{A}$ and $Orb_{\mathcal{A}}(i)$ be the orbit of the row $a'_i \in \mathbb{A}'$ under the action of the group $\mathcal{A}$ over the set $\mathbb{A}'$. Then:*

1) *The group $\mathcal{B} = \rho\mathcal{A}\rho^{-1}$ is the group of automorphism row permutations of the matrix B;*

2) *the orbit $Orb_{\mathcal{B}}(i) = \rho^{-1}Orb_{\mathcal{A}}(\rho_i)$.*

Proof. This proposition follows directly from the homomorphism principle [10] applied on the operartion of the group $S_m(\pi')$ on the row sets of the matrices A and B. $\square$

When a new solution closer to the canonical matrix $c(A)$ has been obtained, this proposition gives an easy way to transform the orbits and stabilizers of the previous matrix $\widetilde{A}_{old}$ into the corresponding orbits and stabilizers of the new matrix $\widetilde{A}_{new}$. This way the search in the tree corresponding to $\widetilde{A}_{new}$ does not start from the beginning, but goes on from the corresponding branch.

Description of the main algorithm.

3.9. This is a recursive *backtrack search* algorithm. By 3.5 the choice set is the group $S_m(\pi')$ of row permutations, which do not change the partition π'. The root of the search tree associated with this group is the empty set. The first level of nodes are the numbers of the rows of the first cell $\widetilde{\pi}'(1)$ of the partition π' (see 2.2). Each j-th level node $j \in \mathbb{N}_m^+$ is an element of the cell $\widetilde{\pi}'(j)$ and corresponds to a partial solution $(i_1, i_2, \ldots, i_j)$ formed by the nodes on the path from the root to the node. The descendants of the node are the elements of the cell $\widetilde{\pi}'(j+1)$ which are not in a previous level of the ancestor branch. They correspond to partial solutions $(i_1, \ldots, i_j, i_{j+1})$ obtained by adding the element $i_{j+1} \in \pi'(j+1) \setminus \{i_1, \ldots, i_j\}$. The leaves of the tree are at level m and correspond to complete solutions.

For a more compact and formal description of the algorithm, we use pseudocode of a syntax close to that of a high level programming language.

3.10. Procedure CANONICAL-FORM (*Step* 1) provides the necessary input data (the matrix A and the partitions π' and π''), initializes the matrix $\widetilde{A}$ (containing the lexicographically greatest matrix, obtained by now) and the set G (containing the obtained full solutions), makes the first call of the recursive procedure SEARCH and outputs the results.

```
0: procedure  CANONICAL-FORM;
1: begin
2:    INPUT $(A, \pi', \pi'')$
3:    $\widetilde{A} := s(A, \pi'')$
4:    $G := \{\}$
5:    SEARCH $(\{\}, 0))$
6:    $c(A) := \widetilde{A}$
7:    OUTPUT $(c(A))$
8: end.
```

Step 1. *Starting procedure.*

3.11. Procedure SEARCH is recursive, it performs search in depth.

```
0: procedure  SEARCH $((i_1, \ldots, i_s)$:partial solution, $s : 0..m)$
1: begin
2:    if $s < m$ then
3:      begin
4:        FORM-CHOICE-SET $((i_1, \ldots, i_s), s)$;
          -forms the choice set $\mathbb{I}_{s+1}$
5:        for all $i_{s+1} \in \mathbb{I}_{s+1}$ do
6:          SEARCH $((i_1, \ldots, i_{s+1}), s + 1)$
7:      end
8:    else
9:      PROCESS-COMPLETE-SOLUTION $((i_1, \ldots, i_s))$
10: end
```

Step 2. *Recursive procedure performing in depth search*

It calls FORM-CHOICE-SET (line 4) and PROCESS-COMPLETE-SOLUTION (line 9).

3.12. Procedure FORM-CHOICE-SET (*Step* 3) gets as input the partial solution $(i_1, \ldots, i_t)$ and forms the set $\mathbb{I}_{t+1}$ of numbers of rows which can be put in $(t + 1)$-st position of the extended partial solution $(i_1, \ldots, i_t, i_{t+1})$.

```
0: procedure FORM-CHOICE-SET $((i_1,\dots,i_t)$:partial solution, $t: 0..m)$
1: begin
2:    if $t = 0$ then $\mathbb{I}_1 := \pi'(1)$
3:    else
4:       if $\widetilde{A}_{(1,\dots,t)} > s(\widetilde{A}_{(i_1,\dots,i_t)}, \pi'')$ then
5:          $\mathbb{I}_{t+1} := \emptyset$
6:       else
7:          begin
8:             $\eta'' :=$ FINER-PARTITION $((i_1,\dots,i_t), \pi'')$
               -$\eta''$ is a finer partition of the columns
9:             $\widetilde{\mathbb{I}}_{t+1} := \pi'(t+1) \setminus \{i_1,\dots,i_t\}$
               -$\widetilde{\mathbb{I}}_{t+1}$ is an extended choice set
10:            for all $i \in \widetilde{\mathbb{I}}_{t+1}$ do $Spect[i]$:=SPECT$(i, \eta'')$
11:            $BestSpect := \max\{Spect[i] \mid i \in \widetilde{\mathbb{I}}_{t+1}\}$
12:            $\mathbb{I}_{t+1} := \{i \in \widetilde{\mathbb{I}}_{t+1} \mid Spect[i] = BestSpect\}$
13:         end
14: end
```

Step 3. *Procedure determining the choice set* $\mathbb{I}_{t+1}$.

At the first call of FORM-CHOICE-SET ($t = 0$), the choice set $\mathbb{I}_1 = \widetilde{\pi}'(1)$ is generated, and at any next call – the set $\mathbb{I}_{t+1}$, which (by 3.6) is empty iff the matrix $s(\widetilde{A}(i_1,\dots,i_t),\pi'')$ is smaller than the matrix of the first t rows of $\widetilde{A}$.

When $\widetilde{A}(1,\dots,t) \le s(\widetilde{A}(i_1,\dots,i_t),\pi'')$, the partial solution $(i_1,\dots,i_t)$ has to be extended. How is $\mathbb{I}_{t+1}$ formed?

Line 8 in *Step* 3 defines a new partition η'' of the column set, such that:

1) η'' is finer than π'' ($\eta'' \preccurlyeq \pi''$) and

2) the columns in the different cells of η'' in the matrix $s(\widetilde{A}(i_1,\dots,i_t),\pi'')$ are equal.

This is done by the *function* FINER-PARTITION $((i_1,\dots,i_t),\pi'')$.

3.13. *Line* 9 in *Step* 3 creates an extended choice set $\widetilde{\mathbb{I}}_{t+1} \supseteq \mathbb{I}_{t+1}$. It contains the rows which can be put at position $t+1$ without changing π'. The set consists of all indexes (not in the partial solution $(i_1,\dots,i_t)$) of rows of the cell of the partition π' which contains row a'_{t+1}, i.e. $\widetilde{\mathbb{I}}_{t+1} = \pi'(t+1)\setminus\{i_1,\dots,i_t\}$.

For each $i \in \widetilde{\mathbb{I}}_{t+1}$ a vector is formed, which is called *spectrum of the row* a'_i *under the partition* η'' and denoted by *Spect*(i). The vector is obtained from the i-th row of the matrix $\widetilde{A}$, by sorting of the elements $a_{i,j}$ in descending order in the different cells of η'' (function SPECT(i,η''), *line* 10 of *Step* 3). The greatest of all spectra thus obtained is chosen and the numbers of the rows with this spectrum form the set $\mathbb{I}_{t+1}$ (*lines* 11 and 12 of *Step* 3). As similar effect to that of the spectrum usage is achieved by McKay [22] and Bouyukliev [4] by using invariants suitable for binary matrices. The spectrum is defined and effective for any integer matrix.

Remark. When the partition η'' is discrete, the choice sets $\mathbb{I}_u$ for $t < u \leq m$ are cells of a discrete partition of the set of the rows which remained outside the partial solution. In that case the algorithm can go on to sort these rows in lexicographically descending order without changing the partitions π' and η''.

3.14. The procedure PROCESS-COMPLETE-SOLUTION processes the complete solutions and the data they hold. Two types are offered.

First type. It is useful with small matrices, matrices with small automorphism groups, or matrices for which the starting partition π' is of relatively small cells. If not, it is quite slow, and in some cases it cannot solve the problem in reasonable time (for instance filtering away of equivalent Hadamard matrices of order 64 [20]). This type finds in addition all automorphisms of the group $Aut(c(A), S)$, i.e, the full group of automorphisms of the canonical matrix $c(A)$ preserving π' and π''.

```
procedure PROCESS-COMPLETE-SOLUTION $((i_1, \dots, i_m) :$ compl solution$)$
 1: begin
 2:   if $s(\widetilde{A}(i_1, \dots, i_m), \pi'') > \widetilde{A}$ then
 3:     begin
 4:       $\widetilde{A} := s(\widetilde{A}(i_1, \dots, i_m), \pi'')$
 5:       $G := (i_1, \dots, i_m)G(i_1, \dots, i_m)^{-1}$
          -transforms the set of automorphisms $G$ by 3.8
 6:       for all $t \in \mathbb{N}_m^+$ do
 7:         $\mathbb{I}_t := (i_1, \dots, i_m)\mathbb{I}_t$
            -transforms the choice sets.
 8:     end
 9:   else
10:     if $s(\widetilde{A}(i_1, \dots, i_m), \pi'') = \widetilde{A}$ then
        -$(i_1, \dots, i_m)$ is a new automorphism of $\widetilde{A}$
11:       $G := G \bigcup (i_1, \dots, i_m)$
12: end
```

Step 4. *Procedure for processing of each complete solution* (*Type* 1)

Consider now the changes of the set G of the automorphisms of $\widetilde{A}$ which have been currently found.

If $s(\widetilde{A}(i_1, \dots, i_m), \pi'') > \widetilde{A}$, then the matrix $\widetilde{A}$ must be changed (line 4). As a result the set G will contain the automorphisms of the previous matrix $\widetilde{A}$, not of the new one. That is why line 5 transforms it by 3.8.

If $s(\widetilde{A}(i_1,\dots,i_m),\pi'') = \widetilde{A}$, the current canonical matrix $\widetilde{A}$ is not changed, and the permutation $(i_1,\dots,i_m)$ is its automorphism and is added to G (line 12).

3.15. ***Second type.***

```
0: procedure PROCESS-COMPLETE-SOLUTION ($(i_1,\dots,i_m)$ : complete solution)
1:  begin
2:    if $s(\widetilde{A}(i_1,\dots,i_m),\pi'') > \widetilde{A}$ then
3:      begin
4:        $\widetilde{A} := s(\widetilde{A}(i_1,\dots,i_m),\pi'')$
5:        $G := (i_1,\dots,i_m)G(i_1,\dots,i_m)^{-1}$
          -transforms the set of automorphisms $G$
            by 3.8, i.e.  $G_{new} := (i_1,\dots,i_m)G_{old}(i_1,\dots,i_m)^{-1}$
6:        $\Pi_G := (i_1,\dots,i_m)\Pi_G$
          -transforms the orbits of the group with generating set  $G_{old}$
          -into the corresponding orbits of the group generated by  $G_{new}$
           by 3.8, i.e.  $\Pi_{G_{new}} := (i_1,\dots,i_m)\Pi_{G_{old}}$
7:        for all $t \in \mathbb{N}_m^+$ do
8:          $\mathbb{I}_t := (i_1,\dots,i_m)\mathbb{I}_t$
          -transforms the choice sets
9:      end
10:   else
11:     if $s(\widetilde{A}(i_1,\dots,i_m),\pi'') = \widetilde{A}$ then
          -$(i_1,\dots,i_m)$ is a new automorphism of $\widetilde{A}$
12        begin
13:         $G := G \bigcup (i_1,\dots,i_m)$
14:         MODIFY-ORBITS-AND-CHOICE-SETS
15        end
16: end
```

Step 5. *Procedure for processing of each complete solution* (*Type* 2)

By Type 2. When a new automorphism permutation of the rows of $\widetilde{A}$ is found, the orbits of the group are determined with respect to the new generating set G (*line* 13 of *Step* 5). As a result the number of orbits might become smaller as some of them might be united. Then MODIFY-ORBITS-AND-CHOICE-SETS checks the choice sets $\mathbb{I}_1,\dots,\mathbb{I}_m$ and if some of them contains more than one representative of one and the same cell of the partition Π'_G, only the smallest one is left. This reduction of the choice sets leads to substantial improvement of the computation speed. Meringer [15] achieves similar effect of cutting off some search tree branches by maintaining a sequence of automorphism groups that contain each other.

3.16. Let us illustrate the work of the algorithm by an example.
▼ **Example.** *Let*

$$A = \begin{pmatrix} 1\,0\,1\,1\,1\,0\,0\,0\,0\,1 \\ 0\,0\,1\,1\,0\,1\,1\,0\,1\,0 \\ 1\,1\,0\,1\,0\,0\,0\,1\,1\,0 \\ 0\,0\,0\,0\,1\,0\,1\,1\,1\,1 \\ 0\,1\,1\,0\,0\,1\,0\,1\,0\,1 \\ 1\,1\,0\,0\,1\,1\,1\,0\,0\,0 \end{pmatrix}$$

be the incidence matrix of a 2-(6,3,2) *design.*

We want to find the canonical form of the matrix A under the operation of the group of all matrix permutations, which fix its first row (corresponds to the group of automorphisms of the design which fix its first point).

We start with partitions $\pi' = \{\{1\}, \{2,3,4,5,6\}\}$ and $\pi'' = \{1,2,\ldots,10\}$. The group S of matrix permutations preserving these partitions has 5!10! elements. We will follow the work of the algorithm in details.

1) *Initialization* – by CANONICAL-FORM ($t = 0$). At the beginning $c(A)$ is

$$\widetilde{A} = s(A, \pi'') = \begin{pmatrix} 1\,1\,1\,1\,1\,0\,0\,0\,0\,0 \\ 1\,1\,0\,0\,0\,1\,1\,1\,0\,0 \\ 1\,0\,1\,0\,0\,1\,0\,0\,1\,1 \\ 0\,0\,0\,1\,1\,1\,1\,0\,1\,0 \\ 0\,1\,0\,1\,0\,0\,0\,1\,1\,1 \\ 0\,0\,1\,0\,1\,0\,1\,1\,0\,1 \end{pmatrix},$$

and G is the empty set.

2) *Forwards* – first call of SEARCH. The choice set for the first element is $\mathbb{I}_1 = \widetilde{\pi}'(1) = \{1\}$. The first partial solution (1) is chosen and the number 1 is taken out of the choice set $\mathbb{I}_1$, i.e, $\longrightarrow \mathbb{I}_1 = \emptyset$.

To form the choice set of the second element, we consequently define:
► the set $\widetilde{\mathbb{I}}_2 = \{2,3,4,5,6\}$ (of all elements of the cell $\widetilde{\pi}'(2)$, which do not take part in the current partial solution);
► the finer partition $\eta'' = \{1,2,3,4,5\}, \{6,7,8,9,10\}$,
► the vectors $Spect(i) = (1,1,0,0,0,1,1,1,0,0)$, for $i \in \widetilde{\mathbb{I}}_2$.

In this case all rows with numbers in $\widetilde{\mathbb{I}}_2$ have one and the same spectrum under the partition η'', this is why the choice set for the second element of the partial solution is $\mathbb{I}_2 = \{2,3,4,5,6\}$.

3) *Forwards* – second call of SEARCH. The partial solution (1) is extended by adding a second element equal to 2 and then taking 2 out of $\mathbb{I}_2$, i.e,

$\longrightarrow \mathbb{I}_2 = \{3,4,5,6\}$. The matrix

$$s(\widetilde{A}(1,2),\pi'') = \begin{pmatrix} 1\,1\,1\,1\,1\,0\,0\,0\,0\,0 \\ 1\,1\,0\,0\,0\,1\,1\,1\,0\,0 \end{pmatrix}$$

is equal to the matrix $\widetilde{A}(1,2)$, which shows that the partial solution (1,2) has to be extended. The choice set $\mathbb{I}_3$ is defined by consequently finding:
▶ the set $\widetilde{\mathbb{I}}_3 = \{3,4,5,6\}$;
▶ the partition $\eta'' = \{1,2\},\{3,4,5\},\{6,7,8\},\{9,10\}$;
▶ the spectra of the rows of the set $\widetilde{\mathbb{I}}_3$ under the partition η'':
$Spect(3) = Spect(5) = (1,0,1,0,0,1,0,0,1,1)$,
$Spect(4) = Spect(6) = (0,0,1,1,0,1,1,0,1,0)$;
▶ The spectrum of the third and fifth rows is lexicographically greatest and thus $\mathbb{I}_3 = \{3,5\}$

4)*Forwards* – in the way described above the next partial solutions are obtained, namely:
$(1,2,3) \longrightarrow \mathbb{I}_3 = \{5\}, \mathbb{I}_4 = \{5\}$,
$(1,2,3,5) \longrightarrow \mathbb{I}_4 = \emptyset, \mathbb{I}_5 = \{6\}$,
$(1,2,3,5,6) \longrightarrow \mathbb{I}_5 = \emptyset, \mathbb{I}_6 = \{4\}$ and $(1,2,3,5,6,4) \longrightarrow \mathbb{I}_6 = \emptyset$.
The matrix

$$s(\widetilde{A}(1,2,3,5,6,4),\pi'') = \begin{pmatrix} 1\,1\,1\,1\,1\,0\,0\,0\,0\,0 \\ 1\,1\,0\,0\,0\,1\,1\,1\,0\,0 \\ 1\,0\,1\,0\,0\,1\,0\,0\,1\,1 \\ 0\,1\,0\,1\,0\,0\,1\,0\,1\,1 \\ 0\,0\,1\,0\,1\,0\,1\,1\,1\,0 \\ 0\,0\,0\,1\,1\,1\,0\,1\,0\,1 \end{pmatrix},$$

corresponding to the obtained complete solution is closer to the canonical form and is saved as $c(A)$, which leads to changes in:
▶ The matrix $\widetilde{A} = s(\widetilde{A}(1,2,3,5,6,4),\pi'')$;
▶ the set of automorphism row permutations and $G = \{e' = (1,2,3,4,5,6)\}$ and
▶ the choice sets $\mathbb{I}_t, t = \mathbb{N}_6^+$ respectively: $\mathbb{I}_6 = \mathbb{I}_5 = \mathbb{I}_4 = \emptyset$, $\mathbb{I}_3 = \{4\}$, $\mathbb{I}_2 = \{4,5,6\}$, $\mathbb{I}_1 = \emptyset$.

5) *Backwards* – it goes back until a nonempty choice set is found, i.e, until $\mathbb{I}_3 = \{4\}$ and the partial solution is $(1,2)$.

6) *Forwards* – as described above until a new complete solution is found. The matrix corresponding to it $s(\widetilde{A}(1,2,4,3,6,5),\pi'')$ is equal to the matrix $\widetilde{A}$, which means that the permutation $\alpha_1 = (1)(2)(3,4)(5,6)$ is an automorphism $\widetilde{A}$. The numbers 4 and 6 are taken out of the only nonempty choice set $\mathbb{I}_2 =$

$\{3,4,5,6\}$ because they are in one and the same orbit with smaller numbers. The choice sets $\mathbb{I}_3, \mathbb{I}_4, \mathbb{I}_5$ and $\mathbb{I}_6$ are empty, and thus the group with generating set $G = \{e', \alpha_1\}$ is the full group of automorphisms fixing the first two rows of the matrix $\widetilde{A}$, and the partition corresponding to it is $\Pi_G = \{\{1\},\{2\},\{3,4\},\{5,6\}\}$.

7) *Backwards* – until partial solution (1) and $\mathbb{I}_2 = \{3,5\}$.

8) *Forwards* – until the next complete solution (1,3,2,5,4,6). It is an automorphism $\alpha_2 = (1)(2,3)(4,5)(6)$ of $\widetilde{A}$. The group of automorphisms generated by $G = \{e', \alpha_1,\ \alpha_2\}$ defines partition $\Pi_G = \{\{1\}\{2,3,4,5,6\}\}$. Taking into account that 3 and 5 are in one and the same cell of this partition, the choice set $\mathbb{I}_2 = \{5\}$ is reduced to the empty set. Thus all choice sets are empty, and the algorithm stops its work.

As a result we obtain that the canonical form of the incidence matrix under the group of matrix permutations $S_6(\pi') \times S_{10}(\pi'')$ is

$$c(A) = \widetilde{A} = \begin{pmatrix} 1\,1\,1\,1\,1\,1\,0\,0\,0\,0 \\ 1\,1\,0\,0\,0\,1\,1\,1\,0\,0 \\ 1\,0\,1\,0\,0\,1\,0\,0\,1\,1 \\ 0\,1\,0\,1\,0\,0\,1\,0\,1\,1 \\ 0\,0\,1\,0\,1\,0\,1\,1\,1\,0 \\ 0\,0\,0\,1\,1\,1\,0\,1\,0\,1 \end{pmatrix}$$

and the group $Aut(c(A), S_m(\pi'))$ of automorphisms fixing the first row has generating set $G = \{e', \alpha_1, \alpha_2\}$.

4. Combinatorial design problems.

For the basic concepts and notations concerning combinatorial designs, refer, for instance, to [1] or [30].

Balanced Incomplete Block Design (BIBD) with parameters 2-(v,b,r,k,λ) (2-(v,k,λ) design or 2-design for short) is a pair $\{V,B\}$ where $V = \{P_1, P_2, \dots, P_v\}$ is a finite set of $v > k \geq 2$ elements (called *points*) and $B = \{B_1, B_2, \dots, B_b\}$ is a collection of b k-element subsets (called *blocks*), such that each pair of points of V is contained in exactly λ blocks of B. The parameters of a 2-design must satisfy the following conditions

$$b.k = v.r, \qquad r.(k-1) = \lambda.(v-1) \qquad \text{and} \qquad b \geq v (\textit{Fisher's inequality}).$$

An incidence matrix of a 2-(v,b,r,k,λ) design is a $(0,1)$ matrix with v rows and b columns, where the element of the i-th row ($i = 1,2,\dots,v$) and j-th column ($j = 1,2,\dots,b$) is 1 if the i-th point of V occurs in the j-th block of B ($P_i \in B_j$) and 0 otherwise. The design is completely determined by its incidence matrix.

Isomorphic BIBDs. Two designs are *isomorphic* if there exists a one-to-one correspondence between the point and block sets of the first design and the point and block sets of the second design, and if this one-to-one correspondence does not change the incidence. The incidence matrices of two isomorphic designs are equivalent.

Automorphism of a BIBD. An *automorphism* is an isomorphism of the design to itself, i.e, a permutation of the points that transforms the blocks into blocks. The set of all automorphisms of a design forms a group called the *full group of automorphisms.* Each subgroup of this group is a group of automorphisms of the design.

Orbit Matrix of a BIBD with automorphism of order p. Let D(V,B) be a 2-(v, k, λ) design with an automorphism of a prime order p. Without loss of generality we can assume that the automorphism $\varphi = (\rho, \sigma)$ acts on the points by

$$\rho = (1, \dots, p)(p + 1, \dots, 2p) \dots ((mp - p + 1, \dots, mp)(mp + 1) \dots (v)$$

and on the blocks by

$$\sigma = (1, \dots, p)(p + 1, \dots, 2p) \dots ((np - p + 1, \dots, np)(mnp + 1) \dots (b)$$

and it partitions the points and blocks of the design into fixed (trivial) and non-fixed orbits. The $((m + v - mp) \times (n + b - np))$ matrix S with entries equal to the number of points of the i-th point orbit in any block of the j-th block orbit is called *orbit matrix of the design D with respect to the automorphism* φ.

Local approach. This is a method of generating 2-designs with an automorphism of prime order p. By this method all the possible non-equivalent orbit matrices are formed first, and then these matrices are extended to incidence matrices of designs.

Here we present the algorithm applications for orbit matrices and incidence matrices of combinatorial designs.

4.1. Canonical form of orbit matrix. Let the $((m + f) \times (n + h))$ matrix A be an orbit matrix of a 2-(v, k, λ) design with an automorphism of prime order p with f fixed points and h fixed blocks. Most commonly all non-fixed point (block) orbits of A correspond to mutually interchangeable rows (columns) of A and define the first cell, and well as all fixed point (block) orbits define the other cell of the points (blocks) partition π' (π''), i.e, if $f > 0$ the partition π' has two cells, if $f = 0$ it has one, if $h > 0$ the partition π'' has two cells and if $h = 0$ it has one.

Let the rows and columns of the matrix A be ordered in such a way that the first m rows (n columns) correspond to non-fixed point (block) orbits and the next f rows (h columns) to the fixed orbits. The set of all possible permutations of the rows which preserve the partition π' is $S_{m+f}(\pi') = S_m \times S_f$ and the set of all possible permutations of the columns that preserve the partition π'' is $S_{n+h}(\pi'') = S_n \times S_h$. The set $G = S_{m+f}(\pi') \times S_{n+h}(\pi'')$ is the complete group of matrix permutations of A. If in the given task for constructing the 2-design there are no additional limitations considering the type of the possible automorphism, we will work with the group G. In other cases we will work with sub-groups of G derived from sub-groups of $S_{m+f}(\pi')$ and $S_{n+h}(\pi'')$, and respectively of the symmetric groups S_m, S_f, S_n, S_h.

4.2. Equivalence test of orbit matrices. According to 3.2 if G is a group of matrix permutations and the matrices A and B are G-equivalent then their canonical forms under the operation of G, are equal, i.e, if $A \overset{G}{\sim} B$ then $c(A) = c(B)$.

This allows us to establish G-equivalence of two orbit matrices by comparing their G-canonical forms. When they are equal, then the matrices are G-equivalent and otherwise the matrices are inequivalent.

4.3. Isomorphism test of designs. In a similar way, we can perform an isomorphism test of designs by comparing the canonical forms of their incidence matrices according to the action of the group $S_v \times S_b$.

4.4. Full automorphism group of the orbit matrix. Let A be an orbit matrix of a 2-design and G the related group of matrix permutations. We can find the full group $Aut(c(A)) \leq G$ of automorphisms of the matrix A in its canonical form under the action of G, or only its generating set (see *Step*.5).

4.5. Full automorphism group of a design. To find the full group of automorphisms of a design, we find the group of automorphism row permutations of the canonical form of the incidence matrix of the design under the trivial partitions $\pi' = \{1, \ldots, v\}$ and $\pi' = \{1, \ldots, b\}$.

5. Applications.

The algorithms described in this work were used in parallel with programs described in [29] for the classification up to isomorphism, and for the computing the order of the automorphism group of 2-(15,7,6) designs with automorphisms of orders 7 and 5 [16], 2-(15,7,6) designs with automorphisms of order 3 [18], 2-(19,9,8) designs with automorphisms of order 3 [19], Hadamard 2-(63,31,15) designs invariant under the dihedral group of order 10 [17], and to compute the order of automorphism groups needed for the classification of the line spreads of PG(5,2)[20].

REFERENCES

[1] Beth Th., D. Jungnickel, H. Lenz. Design Theory, Cambridge University Press, 1993.

[2] Bosma W., J. Cannon (Eds) Handbook of Magma Functions, Edition 2.13 (2006), 4350 pages.

[3] Boukliev I. 'Q – EXTENSION'– strategy in algorithms, Proceedings of the International Workshop ACCT, Bansko, Bulgaria, 2000, 84–89.

[4] Boukliev I. Algorithmic approaches to the study of linear codes, Doctor of sciences thesis, Institute of Mathematics and Informatics, BAS, Sofia, Bulgaria, 2008 (in Bulgarian).

[5] Carter J. On the Existence of a Projective Plane of Order Ten, PhD Thesis, Univ. of Calif., Berkeley, 1974.

[6] Denny P., P. B. Gibbons. Case studies and new results in combinatorial enumeration *J. Combin. Des.*, 2000, 239–260.

[7] Denny P. C., R. Mathon. A census of $t-(t+8, t+2, 4)$ designs, $2 \leq t \leq 4$, *J. Statist. Plann. Inference*, **106** (2002) 5–19.

[8] The GAP Group, *GAP – Groups, Algorithms, and Programming, Version 4.4.10*; 2007, (http://www.gap-system.org).

[9] Grund R., A.Kerber, R.Laue. MOLGEN, ein Computeralgebra–System für die Konstruktion molekularer Graphen. *Communications in mathematical chemistry* (Match) **27** (1992), 87–131.

[10] Grüner T., R.Laue, M.Meringer. Algorithms for group actions: homomorphism principle and orderly generation applied to graphs, *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, **28** (1997), 113–122.

[11] Kapralov S. Algorithms for generation and study of orbit matrices, Scientific conference dedicated to the 100 anniversary of academic Lubomir Chakalov, 14–15.02.1986 (in Bulgarian).

[12] Kaski P. Algorithms for classification of combinatorial objects, Research Report A94, Helsinki University of Technology, Laboratory for Theoretical Computer Science, Espoo, Finland, June 2005.

[13] Krčadinac V. Construction and classification of finite structures by computer. PhD thesis, May 2004.

[14] Kaski P., P. R. Ostergard. Classification Algorithms for Codes and Designs, Springer-Verlag, Berlin Heidelberg, 2006.

[15] Meringer M. Erzeugung regularer Graphen. Master's thesis, Universitat Bayreuth, January 1996.

[16] Mateva Z., S. Topalova. Enumeration of 2-(15,7,6) designs with automorphisms of order 7 or 5. *Mathematics and Education in Mathematics* **31** (2006), 270–274.

[17] Mateva Z., S. Topalova. Hadamard 2-(63,31,15) Designs Invariant under the Dihedral Group of Order 10, International Conference Pioneers of Bulgarian Mathematics, Book of Abstracts, Sofia, July 8–10, 2006, 77.

[18] Mateva Z., S. Topalova. Quasidoubles of Hadamard 2-(15,7,3) Designs with Automorphisms of Order 3, *Mathematics and Education in Mathematics* **32** (2007), 180–185.

[19] Mateva Z., S. Topalova. Doubles of Hadamard 2-(19,9,4) Designs with Automorphisms of Order 3. In: Proceedings of the Fifth International Workshop on Optimal Codes and Related Topics, Balchik, June 2007, Bulgaria.

[20] Mateva Z., S. Topalova. Line spreads of PG(5,2), *J. Comb. Des.*, submitted.

[21] Meringer M. Fast generation of regular graphs and construction of cages, *J. Graph Theory* **30** (1999) No. 2, 137–146.

[22] McKay B. Nauty user's guide (version 1.5), Technical Report TR-CS-90-02, Computer Science Department, Australian National University, 1990.

[23] McKay B. Practical graph isomorphism. Congressus Numeration, **30** (1981), 45–87.

[24] Overton M., A.Proskurowski. Canonical incidence matrices of graphs, *Springer Netherlands*, Vol. **19**, No 2 (June, 1979), 271–273.

[25] Kaski P., P.R.J.Ostergard. Classification Algorithms for Codes and Designs, Springer, Berlin, 2006.

[26] Ostergard P. Classification of binary/ternary one-error-correcting codes, *J. Discrete Mathematics* **223** (2000), 253–262.

[27] Soicher L. The GRAPE package for GAP, Version 4.3, 2006, (`http://www.maths.qmul.ac.uk/~leonard/grape/`) .

[28] Soicher L. The DESIGN package for GAP, Version 1.3, 2006, (`http://designtheory.org/software/~gap_design/`).

[29] Topalova S. Construction and study of combinatorial designs with predefined automorphisms, Ph.D. Thesis, Institute of Mathematics and Informatics, Sofia, 1997 (in Bulgarian).

[30] Tonchev V. Combinatorial configurations, Longman Scientific and Technical, New York, 1988.

[31] Walker R. An Enumerative Technique for a Class of Combinatorial Problems, *Proc. AMS Symp. Appl. Math.*, Vol. **X** (1960), 91–94.

Zlatka Teneva Mateva
Department of Mathematics
Technical University
Varna, Bulgaria
e-mail: `ziz@abv.bg`

Received December 1, 2008
Final Accepted January 28, 2009