Mathematica Pannonica
24/2 (2013), 183–196

FINDING UNITALS IN SYMMETRIC DESIGNS USING MODIFIED GENETIC ALGORITHM

Dean **Crnković** and Doris **Dumičić Danilović**

Department of Mathematics, University of Rijeka, Radmile Matejčić 2, 51000 Rijeka, Croatia

Received: January 2013

MSC 2010: 05 B 05, 90 C 59

Keywords: Symmetric design, unital, genetic algorithm.

Abstract: We present an algorithm for finding unitals as subdesigns of symmetric designs. Further, we employ this algorithm to find unitals in symmetric (66,26,10) designs. This problem is here formulated as a combinatorial optimization problem whose solutions are binary matrices. Using the modified genetic algorithm (MGA) we have found eleven new $2-(11,5,10)$ designs (unitals) in the symmetric (66,26,10) design admitting an automorphism of order 55. In addition, we have found 63 new unitals in the symmetric (36,15,6) design with the full automorphism group of order 42.

1. Introduction

Genetic algorithms (GA) are search and optimization heuristic population based methods which are inspired by the natural evolution process. In each step of the algorithm, a subset of the whole solution space, called *population*, is being treated. The population consists of individuals – *chromosomes*. Instead of finding an optimal solution within the whole solution space, the algorithm concentrates in optimizing the selected population. Every chromosome represents a possible solution (optimum), which is evaluated using the *fitness function*. In each iteration of the algorithm, a certain number of best-ranked chromosomes (parents)

E-mail addresses: deanc@math.uniri.hr, ddumicic@math.uniri.hr

is selected to create new better individuals (children, offsprings). Offsprings are created by a certain type of recombination (crossover) and they replace the worst-ranked chromosomes of the population, providing convergence to the local optimum. After the offsprings are obtained, a mutation operator is allowed to occur (for the purpose to escape from a local optimum) and the next generation of the population is created. The process is iterated until the evolution condition terminates. This method has presented itself as very efficient for solving a variety of optimization problems, including those NP-hard (see [2]), as well as problems where a feasible solution is only the optimal solution – as it is in the case of combinatorial design construction. The application of metaheuristics seems to be more appropriate to attack larger problem instances. This paper provides some steps in this direction.

A *balanced incomplete block design* $\mathcal{D}$ (BIBD) with parameters 2–(v, k, λ) is a finite incidence structure $(\mathcal{P}, \mathcal{B}, \mathcal{I})$, where $\mathcal{P}$ and $\mathcal{B}$ are disjoint sets and $\mathcal{I} \subseteq \mathcal{P} \times \mathcal{B}$, with the following properties:

1. $|\mathcal{P}| = v$ and $1 < k < v - 1$,
2. every element (block) of $\mathcal{B}$ is incident with exactly k elements (points) of $\mathcal{P}$,
3. every pair of points in $\mathcal{P}$ is incident with exactly λ blocks of $\mathcal{B}$.

A standard way of representing a BIBD with parameters (v, k, λ) is in terms of its incidence matrix $M \equiv \{m_{ij}\}_{b\times v}$ which is a $b \times v$ binary matrix where b and v are the number of blocks and points respectively, such that $m_{ij} = 1$ if the point P_j and block x_i are incident and $m_{ij} = 0$ otherwise. Incidence matrix M has exactly r ones per column, k ones per row and a scalar product of λ between any pair of distinct column. The five parameters defining a $\langle v, b, r, k, \lambda\rangle$-BIBD are related and satisfy the following two relations: $bk = vr$ and $\lambda(v - 1) = r(k - 1)$. Thus parameters b and r are given in terms of the other parameters:

$$b = \frac{v(v-1)\lambda}{k(k-1)}, \qquad r = \frac{(v-1)\lambda}{k-1}. \tag{1.1}$$

The case $b = v$ represents a special design called *symmetric design*. A direct consequence of the definition of a symmetric design is that $r = k$.

R. Mathon and Tran van Trung (see [7]) have extended the notion of unital from projective planes to arbitrary symmetric designs.

Definition 1.1. Let $\mathcal{D} = (\mathcal{P}, \mathcal{B}, \mathcal{I})$ denote a symmetric (v, k, λ) design. Let $\mathcal{U}$ be a subset of points of $\mathcal{D}$ having the property:

Through each point $p \in \mathcal{U}$ there are $k-1$ blocks of $\mathcal{D}$ meeting $\mathcal{U}$ in α points and exactly one meeting $\mathcal{U}$ in p. Such a point set $\mathcal{U}$ is called a *unital*. A block meeting $\mathcal{U}$ in one point is called a tangent and block meeting $\mathcal{U}$ in α points a secant.

Here we point out two basic properties of unitals in symmetric designs which is given in the following lemmas.

Lemma 1.1. *Let $\mathcal{U}$ be a unital in $\mathcal{D}$. Then*

(i) $n = k - \lambda$ *is a square,* (ii) $\alpha = \sqrt{n} + 1$,

(iii) $|\mathcal{U}| = 1 + (k-1)\frac{\sqrt{n}}{\lambda}$, *in particular* $1 + (k-1)\frac{\sqrt{n}}{\lambda}$ *is an integer,*

(iv) *The points of $\mathcal{U}$ and its secants form a* $2-(1+(k-1)\frac{\sqrt{n}}{\lambda}, \sqrt{n}+1, \lambda)$ *design.*

Lemma 1.2. *Let $\mathcal{U}$ be a set of $u = 1 + (k-1)\frac{\sqrt{n}}{\lambda}$ points in $\mathcal{D}$. Assume that any block of $\mathcal{D}$ meets $\mathcal{U}$ in 1 or $\sqrt{n}+1$ points. Then $\mathcal{U}$ is a unital.*

These lemmas are rather obvious, therefore proofs are omitted. However, proofs can be found in [11].

The design obtained from $\mathcal{U}$ and its secants is called a *unital*. For a detailed explanation and further results on unitals in symmetric designs one can refer to [7]. We shall use this representation of a design $\mathcal{D}$ and a unital $\mathcal{U}$ throughout this work.

It is known that there are at least 590 mutually nonisomorphic symmetric (66,26,10) designs (see [4], [8], [10], [12]). D. Crnković and S. Rukavina (see [4]) have proved that up to isomorphism there are three symmetric (66,26,10) designs with automorphism group Frob_{55} acting with orbit distribution (11, 55). Let us denote them by $\mathcal{D}_1$, $\mathcal{D}_2$ and $\mathcal{D}_3$. For one of them, say $\mathcal{D}_3$, they have proved that up to isomorphism it is the only symmetric (66,26,10) design with automorphism group Frob_{55} acting with orbit distribution (11, 11, 11, 11, 11, 11). Associated orbit structure for this distribution is (see [4]):

OS	11	11	11	11	11	11
11	1	5	5	5	5	5
11	5	5	5	5	5	1
11	5	5	5	5	1	5
11	5	5	5	1	5	5
11	5	5	1	5	5	5
11	5	1	5	5	5	5

Elements in this orbit structure represent the number of points in appropriate point orbit which are incident with the representative of the appropriate block orbit. The design $\mathcal{D}_3$ is isomorphic to the one constructed by Tran van Trung, which is the first symmetric (66,26,10) design ever constructed (see [12]).

R. Mathon and Tran van Trung (see [7]) have pointed out that there are 6 unitals in the design $\mathcal{D}_3$, which one can see from Lemma 1.2 and orbit structure OS. For the same reason it is clear that each of the 6 orbits of points of the corresponding automorphism forms a unital in the designs $\mathcal{D}_1$ and $\mathcal{D}_2$ (see [4]). It would be very difficult to find new possible unitals in a symmetric (66,26,10) designs using exhaustive search because of a large number $\binom{66}{11}$ of combinations of points in designs with this parameters. We have found eleven new unitals in the design $\mathcal{D}_3$ using modified genetic algorithm. The paper is organized as follows: in Sec. 2 we describe the developed algorithm, in Sec. 3 we give the obtained experimental results, in Sec. 4 we give conclusion, while in Appendix A we correct a lemma from a paper by R. Mathon and T. van Trung ([7]).

2. Algorithm for finding unitals in a symmetric design

We have used the *modified steady-state genetic algorithm* (MGA) updating the population in a piecemeal fashion rather than all at one time, which is a variation of a genetic algorithm presented in [13]. We have developed an algorithm with 4-tournament selection and 2-points crossover. The idea is to iteratively generate two new offsprings in one tournament and then reintroduce them directly into the population itself, replacing two of the worst ranked chromosomes in that tournament.

The parameters of the symmetric design $\mathcal{D}$ are v, r, k and λ as described in the previous section. Let us denote by $u = |\mathcal{U}|$ the number of points of unital in the design $\mathcal{D}$ which is obtained in the previous section. Without loss of generality we can denote the point set of the design $\mathcal{D}$ as $\mathcal{P} = \{1, 2, \ldots, v\}$. We shall represent each chromosome of our population as a $b \times u$ binary matrix while its columns are copies of columns of the incidence matrix M of the symmetric design $\mathcal{D}$. These columns are called the *genes* of the chromosomes. We shall denote by POP the number of chromosomes in the population. The initial population is generated randomly, namely genes of chromosomes represent the columns that have been selected randomly from the incidence matrix M. A chromosome

shall represent a unital in a symmetric design if through each point $p \in \mathcal{U}$ there are $k-1$ secants and exactly one tangent, as considered in Def. 1.1.

The fitness function measures the deviation of chromosome from the property in Def. 1.1. Let us denote the number of tangents and secants through each point $p_i \in \mathcal{U}$ by t_i and s_i respectively, for $i = 1, 2, ..., u$. We define the fitness value of each chromosome measuring the sum of average values of its deviation from required number of tangents and secants at each point. Let us denote by X the set of all chromosomes in the population. The fitness function that will be utilized here is $c: X \to \mathbb{N} \cup \{0\}$,

$$c(x) = \frac{\sum_{i=1}^{u} |t_i - 1|}{u} + \frac{\sum_{i=1}^{u} |s_i - (k-1)|}{u}, \quad \forall x \in X.$$

The minimal (optimal) value of the fitness function is zero. If this value is reached, the searched unital in a symmetric design is found.

In each algorithm iteration, two of the fittest chromosomes out of four randomly selected ones are chosen. We realized 4-tournament selection by selecting four neighboring chromosomes. In our algorithm the positions of chromosomes are mixed before a selection (it is done as permutation of POP elements), providing that neighboring chromosomes in the current iteration do not stay neighbors in the next one, which gives better results.

Selected chromosomes take part as parents in the 2-point crossover at column level. This means that we randomly choose two columns x and y in the parents genes, $1 \leq x \leq y \leq u$ as crossover points. The genes of both offsprings will be copies of the parents genes in the corresponding locations, as depicted in the following example, where assorted integers represent the points of a symmetric (66,26,10) design, namely they represent the indices of columns in its incidence matrix:

$$\begin{bmatrix} 1 & 2 & 3 & 5 & | & \boxed{10} & \boxed{18} & \boxed{29} & \boxed{31} & | & 42 & 53 & 66 \end{bmatrix} \quad \textit{parent 1}$$
$$\begin{bmatrix} 2 & 4 & 6 & 10 & | & \boxed{16} & \boxed{25} & \boxed{31} & \boxed{38} & | & 51 & 61 & 65 \end{bmatrix} \quad \textit{parent 2}$$

$$\begin{bmatrix} 1 & 2 & 3 & 5 & | & \boxed{16} & \boxed{25} & \boxed{31} & \boxed{38} & | & 42 & 53 & 66 \end{bmatrix} \quad \textit{offspring 1}$$
$$\begin{bmatrix} 2 & 4 & 6 & 10 & | & \boxed{10} & \boxed{18} & \boxed{29} & \boxed{31} & | & 51 & 61 & 65 \end{bmatrix} \quad \textit{offspring 2}$$

In addition, two obtained offsprings are checked by correction operator, that checks the possibly repeated genes. In the case of repeated genes, this operator replaces the repeated gene by randomly selected one not present in the considered offspring.

To keep diversity in the population, the mechanism which checkes whether two parents are equal is implemented in the algorithm. In the case of their equality, one parent is replaced by a random one. Besides, the diversity of genes is also provided by two kind of mutation operators to act on the obtained offsprings with probability p_m. Mutation of the offspring is also performed on the random basis. One offspring gets possibly a replaced gene, randomly selected column from the incidence matrix M, while the other child gets possibly two genes replaced. Mutated offsprings replace two the worst ranked chromosomes in a tournament. The mutation operators shall occur with a probability p_m which is the same for all described operators, due to some experimental experience.

To make the algorithm as efficient as possible, we forced the mutation of the parents by replacing one of its genes with randomly selected column from the incidence matrix M and repeating this process for five times for the *parent 1* and four times for the *parent 2*. Mutated parent replaces the previous one only if its fitness value is less than previous (elitism is preserved). That was an optimum number of replacements based on performing several experiments. Because less number of replacements could stuck the algorithm on nearly optimal, but not good enough solution, namely a nearly optimal solution with a small fitness value still needs a lot of changes in its genes. Furthermore, the fitness value for each chromosome in a new generation is calculated and the algorithm is run again from this new generation if no unital is already found or restarting mechanism is triggered. After many generations there is a good hope that the fitness bias in the reproduction of chromosomes will result in a chromosome so fit that it is actually a unital.

A restarting mechanism is introduced to reactivate the search whenever the stagnation takes place. This is done by keeping a fraction $f_{\%}$ of the top ranked chromosomes in the current population, and refreshing the rest of the population with random chromosomes. This procedure is triggered after a certain number of generations with no further improvement of the current best solution of the fitness function. The algorithm sometimes gets stuck in a solution with very small fitness value but with no further improvement in this value. In the case that this happens, we have introduced an operator which generates new random population. Pseudocode of the developed algorithm is presented below.

Algorithm for finding unitals in a symmetric design written in MATLAB:

1: $POP \leftarrow$ desired population size
2: **for** POP times # generate initial population
3: $P \leftarrow P \cup \{$new random individual$\}$
4: **end**
5: c $\leftarrow 0$ # number of iterations
6: **for** each individual $P_i \in P$
7: $f_i \leftarrow$ AssessFitness(P_i)
8: **end**
9: Best $\leftarrow \min(f_i)$
10: **while** Best> 0
11: $P \leftarrow$ mix the positions of chromosomes in current population
12: $i \leftarrow 1$
13: **while** $i < POP$
14: P_a, $P_b \leftarrow$ select two of the fittest between the ith and $(i+3)$th chromosome
15: **if** $P_a == P_b$
16: $P_b \leftarrow$ new random chromosome
17: Children C_a, $C_b \leftarrow$ 2PointCrossover(Copy(P_a), Copy(P_b))
18: C_a, $C_b \leftarrow$ CheckRepetition(C_a, C_b) #replace duplicated column
19: $C_a \leftarrow$ Mutation(C_a), $C_b \leftarrow$ Mutation(C_b)
20: $P_c, P_d \leftarrow C_a, C_b$ # replace the two remaining chromosomes in a tournament
21: $MP_a \leftarrow$ Mutation(P_a), $MP_b \leftarrow$ Mutation(P_b)
22: **if** AssessFitness(MP_a) $<$ AssessFitness(P_a)
23: $P_a \leftarrow MP_a$
24: **end**
25: **if** AssessFitness(MP_b) $<$ AssessFitness(P_b)
26: $P_b \leftarrow MP_b$
27: **end**
28: $i \leftarrow i + 4$
29: **end**
30: **for** each chromosome $P_i \in P$
31: $f_i \leftarrow$ AssessFitness(P_i)
32: **end**
33: $c \leftarrow c + 1$
34: Best $\leftarrow \min(f_i)$
35: **if** there is stagnation in the best fitness value
36: $P \leftarrow$ RestartingOper(P) #if stagnation occurs
37: **end**
38: **end**

We have used the modified steady-state genetic algorithm for several reasons. First, it uses half the memory of a traditional genetic algorithm because there is only one population at a time. Second, the parents stay around in the population (elitism), potentially for a very long time, and thus, this runs to the risk of causing the system to prematurely converge to largely copies of a few highly fit individuals. This is avoided by attempting the mutation of two parents and by checking if parents are identical, if they are, one is randomly generated.

The running time complexity of genetic algorithms depends on the genetic operators, their implementation (which may have a very significant effect on overall complexity), the representation of the individuals and the population, and obviously on the fitness function. The MGA is governed by a number of parameters: population size, muterate p_m, fraction $f_{\%}$, time when restarting mechanisms take place, v, k, λ parameters of designs. Even with appropriate parameters, optimal solutions cannot be guaranteed due to the probabilistic nature of the MGA. Also, there are no universally best parameters for the MGA which can achieve the best result for all problems. Since the state transitions of a genetic algorithm are of probabilistic nature, the deterministic definition of convergence was obviously not appropriate. Therefore the definition of stochastic convergence had to be used (see [1]). Due to its stochastic nature, the running time complexity analysis of a MGA is not an easy task. However, if an algorithm does converge, the analysis of the time limit behavior does not give any hints about the expected time for the solution to be found, far less any precise statement (see [9]). The execution time of a genetic algorithm depends on the number of iterations (generations). In many cases, parameters and the number of iterations in genetic algorithms are decided experimentally.

For finding unitals in the symmetric (66,26,10) design $\mathcal{D}_3$, the MGA produces 4.2 iterations per second on a quad-core CPU (3,2 GHz) with population size of 100, 4-tournament selection, 2-point crossover, muterate $p_m = 0.99$, restarting mechanism with $f_{\%} = 10\%$ of the top ranked chromosomes starting after 90 minutes and complete restarting mechanism starting after 180 minutes of stagnation in the best fitness value. Using the described processor and the mentioned parameters of the MGA, a unital is obtained in average 72 minutes, while the minimal time to get one of them was in approximately 23 minutes, according to the results in Table 1. The objective of that heuristic algorithm was to minimize the overall execution time of the complete exhaustive search for unitals in

a symmetric (66,26,10) design. For such an exhaustive search that analyzes all of the $\binom{66}{11}$ possible combinations, it takes approximately 900000 combinations per minute or approximately 2.3 years on the above mentioned CPU. Hence, the time consumption of the MGA is far less then the complete exhaustive search.

3. Experimental results

The results on number of unitals found in already mentioned design $\mathcal{D}_3$ are presented in Table 1, where we give the number of unitals obtained in this symmetric design *nsol*, the number of non-isomorphic unitals among them *niso*, the minimal, the average number of iterations leading to the first solution and the standard deviation denoted by *nmin*, *navi* and σ respectively. Our algorithm has not found new unitals in already mentioned designs $\mathcal{D}_1$ and $\mathcal{D}_2$. We have used a population size of 40, 60 and 100, while for mutation probability p_m we used the values of 0.9, 0.99 or 1. We set the fraction $f_{\%}$ on 10%.

Table 1

Design	*nsol*	*niso*	*nmin*	*navi*	σ
$\mathcal{D}_3$	17	3	5774	18088,27	21728,99

Solution of the considered problem, namely a unital in a symmetric (66,26,10) design is a $2-(11,5,10)$ design, according to Lemma 1.1. It has $b = 55$ blocks and $r = 25$, according to the relations (1.1). The results, namely eleven new unitals in the design $\mathcal{D}_3$, denoted by $U_7, \ldots, U_{17}$, were found and they are represented by the points of $\mathcal{D}_3$ in Table 2.

Table 2: 17 unitals found in $\mathcal{D}_3$ using MGA

Unitals	Points										
U_1	1	2	3	4	5	6	7	8	9	10	11
U_2	12	13	14	15	16	17	18	19	20	21	22
U_3	23	24	25	26	27	28	29	30	31	32	33
U_4	34	35	36	37	38	39	40	41	42	43	44
U_5	45	46	47	48	49	50	51	52	53	54	55
U_6	56	57	58	59	60	61	62	63	64	65	66
U_7	1	2	3	7	9	10	20	31	42	53	64
U_8	1	2	4	5	6	10	12	23	34	45	56
U_9	1	2	6	8	9	11	19	30	41	52	63
U_{10}	1	3	4	5	9	11	22	33	44	55	66
U_{11}	1	3	4	6	7	8	14	25	36	47	58
U_{12}	1	5	7	8	10	11	18	29	40	51	62
U_{13}	2	3	4	8	10	11	21	32	43	54	65
U_{14}	2	3	5	6	7	11	13	24	35	46	57
U_{15}	2	4	5	7	8	9	15	26	37	48	59
U_{16}	3	5	6	8	9	10	16	27	38	49	60
U_{17}	4	6	7	9	10	11	17	28	39	50	61

One can check, using GAP (see [5]), that up to isomorphism there are only three unitals in the design $\mathcal{D}_3$. Furthermore, mutually isomorphic unitals in the design $\mathcal{D}_3$ are given in Table 3.

Table 3: Mutually isomorphic unitals and the order of the associated full automorphism group

Isomorphic unitals in $\mathcal{D}_3$	$\lvert Aut(G)\rvert$
U_1	660
$U_2 \cong U_3 \cong \cdots \cong U_6$	55
$U_7 \cong U_8 \cong \cdots \cong U_{17}$	100

In the symmetric (36,15,6) design $\mathcal{D}'$ with the full automorphism group of order 42 which is isomorphic to the group $\mathrm{Frob}_{21} \times \mathrm{Z}_2$ (see [3]), we have found 66 unitals using the MGA. Among them only three were known before (see [7]). The results on number of unitals found in the design $\mathcal{D}'$ are presented in Table 4. We used a population size of 100, while for mutation probability p_m we used the value of 0.99. We set the fraction $f_{\%}$ on 10%.

A unital in a symmetric (36,15,6) design is a $2-(8,4,6)$ design, according to Lemma 1.1. It has $b = 28$ blocks and $r = 14$. The results, namely 66 unitals in the design $\mathcal{D}'$, denoted by $U'_1, \ldots, U'_{66}$, were found and they are represented by the points of $\mathcal{D}'$ in Table 5.

Table 4

Design	*nsol*	*niso*	*nmin*	*navi*	σ
$\mathcal{D}'$	66	6	4	1660,83	1958,15

One can check, using GAP (see [5]), that up to isomorphism there are exactly six unitals in the design $\mathcal{D}'$. Furthermore, the information on mutually isomorphic unitals in the design $\mathcal{D}'$ are given in Table 6.

Those 66 unitals are all unitals in the design $\mathcal{D}'$. This is checked by a direct search which is performed very fast, because of relatively small number of all possible combinations, which is $\binom{36}{8} = 30260340$.

Performing several experiments, it was noticed that the algorithm regularly progresses close to the solution very fast, but after that, it takes a lot of time to make only slight enhancements. This characteristic of GA has been noticed in case of other combinatorial problems (see [6]).

Table 5: 66 unitals found in $\mathcal{D}'$ using MGA

Unitals	Points							
U'_1	1	16	17	18	19	20	21	22
U'_2	1	16	17	21	23	24	28	32
U'_3	1	16	18	19	23	25	26	34
U'_4	1	16	20	22	23	27	29	31
U'_5	1	17	18	22	24	25	29	33
U'_6	1	17	19	20	24	26	27	35
U'_7	1	18	20	21	25	27	28	36
U'_8	1	19	21	22	26	28	29	30
U'_9	1	23	24	25	26	27	28	29
U'_{10}	1	30	31	32	33	34	35	36
U'_{11}	2	3	4	6	16	33	35	36
U'_{12}	2	3	4	7	19	20	22	25
U'_{13}	2	3	5	8	22	32	34	35
U'_{14}	2	3	6	8	18	19	21	24
U'_{15}	2	3	7	11	16	17	21	32
U'_{16}	2	3	7	11	25	26	27	29
U'_{17}	2	4	5	6	17	21	22	27
U'_{18}	2	4	5	13	16	18	19	34
U'_{19}	2	4	5	13	24	27	28	29
U'_{20}	2	4	7	8	21	31	33	34
U'_{21}	2	5	6	7	19	31	32	36
U'_{22}	2	5	7	8	17	18	20	23
U'_{23}	2	6	8	10	16	20	22	31
U'_{24}	2	6	8	10	24	25	26	28
U'_{25}	2	12	14	15	16	17	18	20
U'_{26}	2	12	14	15	26	28	29	30
U'_{27}	3	4	5	7	17	30	34	36
U'_{28}	3	4	5	8	16	20	21	26
U'_{29}	3	4	8	12	17	18	22	33
U'_{30}	3	4	8	12	23	26	27	28
U'_{31}	3	5	6	7	16	18	22	28
U'_{32}	3	5	6	14	17	19	20	35
U'_{33}	3	5	6	14	23	25	28	29
U'_{34}	3	6	7	8	20	30	32	33
U'_{35}	3	9	13	15	17	18	19	21
U'_{36}	3	9	13	15	23	27	29	31
U'_{37}	4	5	6	8	18	30	31	35
U'_{38}	4	6	7	8	16	17	19	29
U'_{39}	4	6	7	15	18	20	21	36
U'_{40}	4	6	7	15	23	24	26	29
U'_{41}	4	9	10	14	18	19	20	22
U'_{42}	4	9	10	14	23	24	28	32
U'_{43}	5	7	8	9	19	21	22	30
U'_{44}	5	7	8	9	23	24	25	27
U'_{45}	5	10	11	15	16	19	20	21
U'_{46}	5	10	11	15	24	25	29	33
U'_{47}	6	9	11	12	17	20	21	22
U'_{48}	6	9	11	12	23	25	26	34
U'_{49}	7	10	12	13	16	18	21	22
U'_{50}	7	10	12	13	24	26	27	35
U'_{51}	8	11	13	14	16	17	19	22
U'_{52}	8	11	13	14	25	27	28	36
U'_{53}	9	10	11	13	23	33	35	36
U'_{54}	9	10	11	14	18	26	27	29
U'_{55}	9	10	12	15	29	32	34	35
U'_{56}	9	10	13	15	17	25	26	28
U'_{57}	9	11	12	13	20	24	28	29
U'_{58}	9	11	14	15	28	31	33	34
U'_{59}	9	12	13	14	26	31	32	36
U'_{60}	9	12	14	15	16	24	25	27
U'_{61}	10	11	12	14	24	30	34	36
U'_{62}	10	11	12	15	19	23	27	28
U'_{63}	10	12	13	14	21	23	25	29
U'_{64}	10	13	14	15	27	30	32	33
U'_{65}	11	12	13	15	25	30	31	35
U'_{66}	11	13	14	15	22	23	24	26

Table 6: Mutually isomorphic unitals and the order of the associated full automorphism group

Isomorphic unitals in $\mathcal{D}'$	$\lvert Aut(G)\rvert$
U'_{10}	21
$U'_1 \cong U'_9$	336
$U'_2 \cong U'_3 \cong \cdots \cong U'_8$	6
$U'_{11} \cong U'_{13} \cong U'_{20} \cong U'_{21} \cong U'_{27} \cong U'_{34} \cong U'_{37} \cong U'_{53} \cong U'_{55} \cong U'_{58} \cong U'_{59} \cong U'_{61} \cong U'_{64} \cong U'_{65}$	6
$U'_{12} \cong U'_{14} \cong U'_{17} \cong U'_{22} \cong U'_{28} \cong U'_{31} \cong U'_{38} \cong U'_{54} \cong U'_{56} \cong U'_{57} \cong U'_{60} \cong U'_{62} \cong U'_{63} \cong U'_{66}$	3
$U'_{15} \cong U'_{16} \cong U'_{18} \cong U'_{19} \cong U'_{23} \cong U'_{24} \cong U'_{25} \cong U'_{26} \cong U'_{29} \cong U'_{30} \cong U'_{32} \cong U'_{33} \cong U'_{35} \cong U'_{36} \cong U'_{39} \cong \ldots \cong U'_{52}$	6

4. Conclusion

The application of the genetic algorithm to find unitals in a symmetric design has resulted in very encouraging and positive results. It is hard to imagine that an exhaustive search for the 11 points and 55 blocks of symmetric (66,26,10) design that form a unital, namely $2-(11,5,10)$ design, would finish successfully as the size of the search space is definitely extremely large, although the parameters of designs do not seem to be so big. Our algorithm can give an answer to the existence question of unitals in a symmetric design, but, in general, can not give an answer how many unitals are in a symmetric design with particular parameters.

5. Appendix A – Correction of the paper [7] by R. Mathon and T. van Trung

In the paper [7] by R. Mathon and T. van Trung we have found an error in Lemma 5.2, there is λ missing in the denominator of the fraction $\frac{k-1}{2}$, which is obvious from Lemma 5.1 and Lemma 2.1 in that paper. Here we give the correct statement of that lemma.

Lemma 5.1. *If a symmetric* $2-(v,k,\lambda)$ *design admits a unitary polarity, then*

$$s = \frac{k-1}{2\lambda}\left(k+1-\frac{\lambda}{\sqrt{n}}\right)$$

is an integer.

So the consequence of this correction of Lemma 5.2 (from the paper [7]) is that we have to correct the table with a list of parameters for symmetric designs of square order for $n \leq 25$ and $k \leq v/2$ together with the information about unitals and unitary polarities (see [7], p. 248). An answer yes in one of the columns: unital, unitary polarity, means there is at least one solution corresponding to that item. If no solution is known, it is indicated by a question mark.

Table 7: R. Mathon and T. van Trung ([7])

n	v	k	λ	**Unital**	**Unitary Polarity**
4	21	5	1	yes	yes, $PG(2,4)$
4	16	6	2	yes, 2.1 1	no, Remark 5.1
4	15	7	3	yes, 2.1 3	no, Lem. 5.2
9	56	11	2	yes, 2.1 4	no, Lem. 5.2
9	45	12	3	yes	yes, Thm. 5.4
9	40	13	4	yes	yes, $PG_2(3,3)$
9	36	15	6	yes	?
9	35	17	8	yes, Thm. 4.1	no, Lem. 5.2
16	273	17	1	yes	yes, $PG(2,16)$
16	154	18	2	?	no, Lem. 5.2
16	115	19	3	?	no, Lem. 5.2
16	96	20	4	?	?
16	85	21	5	yes	no, Lem. 5.2
16	78	22	6	?	no, Lem. 5.2
16	70	24	8	no, Lem. 2.1	
16	66	26	10	yes	no, Lem. 5.2
16	64	28	12	yes, Thm. 4.2	?
16	63	31	15	yes, Thm. 4.1	no, Lem. 5.2
25	651	26	1	yes	yes, $PG(2,25)$
25	352	27	2	?	no, Lem. 5.2
25	253	28	3	?	no, Lem. 5.2
25	204	29	4	?	no, Lem. 5.2
25	175	30	5	?	?
25	156	31	6	yes	yes, $PG_2(3,5)$
25	133	33	8	?	no, Lem. 5.2
25	120	35	10	?	?
25	112	37	12	?	no, Lem. 5.2
25	105	40	15	?	?
25	100	45	20	yes, Thm. 4.2	?
25	99	49	24	yes, Thm. 4.1	no, Lem. 5.2

Table 8: The correction of Table 7

n	v	k	λ	**Unital**	**Unitary Polarity**
4	21	5	1	yes	yes, $PG(2,4)$
4	16	6	2	yes,[6] 2.1 1	no, Lem. 5.1
4	15	7	3	yes,[6] 2.1 3	no, Lem. 5.1
9	56	11	2	yes,[6] 2.1 4	no, Lem. 5.1
9	45	12	3	yes	yes,[6] Thm. 5.4
9	40	13	4	yes	yes, $PG_2(3,3)$
9	36	15	6	yes	no, Lem. 5.1
9	35	17	8	yes,[6] Thm. 4.1	no, Lem. 5.1
16	273	17	1	yes	yes, $PG(2,16)$
16	154	18	2	?	no, Lem. 5.1
16	115	19	3	?	no, Lem. 5.1
16	96	20	4	?	no, Lem. 5.1
16	85	21	5	yes	no, Lem. 5.1
16	78	22	6	?	no, Lem. 5.1
16	70	24	8	no,[6] Lem. 2.1	
16	66	26	10	yes	no, Lem. 5.1
16	64	28	12	yes,[6] Thm. 4.2	no, Lem. 5.1
16	63	31	15	yes, Thm.[6] 4.1	no, Lem. 5.1
25	651	26	1	yes	yes, $PG(2,25)$
25	352	27	2	?	no, Lem. 5.1
25	253	28	3	?	no, Lem. 5.1
25	204	29	4	?	no, Lem. 5.1
25	175	30	5	?	?
25	156	31	6	yes	yes, $PG_2(3,5)$
25	133	33	8	?	no, Lem. 5.1
25	120	35	10	?	no, Lem. 5.1
25	112	37	12	?	no, Lem. 5.1
25	105	40	15	?	no, Lem. 5.1
25	100	45	20	yes, Thm.[6] 4.2	no, Lem. 5.1
25	99	49	24	yes, Thm.[6] 4.1	no, Lem. 5.1

References

[1] BÄCK, T., FOGEL, D. B. and MICHALEWICS, Z.: *Handbook of Evolutionary Computation*, IOP Publishing Ltd, Bristol, UK, 1997.

[2] CORNEIL, D. G. and MATHON, R.: Algorithmic techniques for the generation and analysis of strongly regular graphs and other combinatorial configurations, *Annals of Discrete Mathematics* **2** (1978), 1–32.

[3] CRNKOVIĆ, D.: Symmetric (36,15,6) design having U(3,3) as an automorphism group, *Glasnik Matematički* **34(54)** (1999), 1–3.

[4] CRNKOVIĆ, D. and RUKAVINA, S.: Symmetric (66,26,10) designs having Frob_{55} as an automorphism group, *Glasnik Matematički* **35(55)** (2000), 271–281.

[5] THE GAP GROUP *GAP – Groups, Algorithms and Programming*, Version 4.4.12, available at `www.gap-system.org`

[6] GOLUB, M. and MARTINJAK, I.: Comparison of Heuristic Algorithms for the N-Queen Problem, in: *ITI*, Cavtat, Dubrovnik, 2007.

[7] MATHON, R. and TRAN VAN TRUNG: Unitals and Unitary Polarities in Symmetric Designs, *Designs, Codes and Cryptography* **10** (1997), 237–250.

[8] MATULIĆ-BEDENIĆ, I., HORVATIĆ-BALDASAR, K. and KRAMER, E.: Construction of new symmetric designs with parameters (66,26,10), *Journal of Combinatorial Designs* **3** (1995), 405–410.

[9] OLIVETO, P. S., HE, J. and YAO, X.: Time Complexity of Evolutionary Algorithms for Combinatorial Optimization: A Decade of Results, *International Journal of Automation and Computing* **04(3)** (2007), 281–293.

[10] PAVČEVIĆ, M.-O. and SPENCE, E.: Some new symmetric designs with $\lambda = 10$ having an automorphism of order 5, *Discrete Mathematics* **196** (1999), 257–266.

[11] DE RESMINI, M. J.: On sets of type (m,n) in BIBD's with $\lambda \geq 2$, *Annals of Discrete Mathematics* **14** (1982), 183–206.

[12] TRAN VAN TRUNG: The existence of symmetric block designs with parameters (41,16,6) and (66,26,10), *Journal of Combinatorial Theory A* **33** (1982), 201–204.

[13] WALLIS, W. D.: *Computational and Constructive Design Theory*, Kluwer Academic Publishers, Dordrecht, Boston, 1996.