

# On the size of minimal blocking sets of $Q(4, q)$ , for $q = 5, 7$

J. De Beule, A. Hoogewijs, and L. Storme  
 Department of Pure Mathematics and Computer Algebra,  
 Ghent University, Krijgslaan 281, 9000 Gent, Belgium  
 email: {jdebeule, bh, ls}@cage.ugent.be  
<http://cage.ugent.be/~{jdebeule, bh, ls}>

## Abstract

Let  $Q(2n + 2, q)$  denote the non-singular parabolic quadric in the projective geometry  $PG(2n + 2, q)$ . We describe the implementation in GAP of an algorithm to study the problem of the minimal number of points of a minimal blocking set, different from an ovoid, of  $Q(4, q)$ , for  $q = 5, 7$ .

## 1 Introduction and definitions

“Projective Geometry” is the branch of geometry dealing with the properties and invariants of geometric figures under projection. As such, projective geometry can be considered as a mathematical framework for computer vision in general, and especially image formation in particular. The main areas of application are those in which image formation and/or invariant descriptions between images are important, such as camera calibration, object recognition, scene reconstruction, image synthesis, and the analysis of shadows. Moreover, computer science has returned the favour by providing powerful tools for the visualization and diffusion of mathematics. Java applets, animated GIF images, and linked HTML pages allow us to present mathematics and, most particularly, geometry in a way that was unthinkable twenty years ago (e.g. [20]). In recent years, there has been an increasing interest in finite projective geometry [12], and important applications to practical topics such as coding theory [14] and digital watermarking [9] have made the field even more attractive.

A *projective geometry* is a geometric structure consisting of various types of objects (points, lines, planes, hyperplanes, etc.) and the relations between them which satisfy a set of axioms (see [1]). The relationship between the objects is called *incidence* and describes where elements such as points, lines, planes, . . . , either coincide or not. The incidence relation is meant to be symmetric, so we say that a point is *incident* with a line (the point is on the line) or that a line is *incident* with a point (the line passes through the point).

Finite projective geometry exists in any number of dimensions. In the sequel, we will consider the projective geometry  $PG(N, q)$  of dimension  $N$  over the Galois field  $GF(q)$ . Galois geometry started around 1950 with the work of the celebrated Italian mathematician Beniamino Segre who studied  $N$ -dimensional projective spaces over Galois fields  $GF(q)$  and substructures of these spaces. His research led to what in the past years has been called the *fundamental problems of finite geometry*: (1) the determination of the maximal and/or minimal number of points belonging to a substructure satisfying specific geometric conditions, (2) the classification of those substructures having the optimal minimal or maximal number of elements, and (3) when the minimal number or maximal number of elements of such a substructure is known, the determination of the cardinality of the second smallest or second largest such substructure.

This paper contributes to the solution of one of these fundamental problems by providing an implementation of an algorithm to investigate the minimal number of points of a specific substructure, called *minimal blocking set of a quadric*  $Q(4, q)$  in  $PG(4, q)$ , for the values  $q = 5, 7$ , different from an ovoid.

A projective geometry  $\text{PG}(N, q)$  arises from a vector space  $V = V(N + 1, q)$  of “rank”  $N + 1$  over the Galois field  $GF(q)$  as a collection of *points* which are the rank 1 subspaces of  $V$ , *lines* which are the rank 2 subspaces of  $V$ , *planes* which are the rank 3 subspaces of  $V$ , *i-spaces* which are the rank  $(i + 1)$ -subspaces of  $V$ , *hyperplanes* which are the rank  $N$ -subspaces of  $V$ , together with the relationship *incidence* defined by containment of the corresponding subspaces. Note that we use the word “rank” for the algebraic dimension of the vector space and use the word “dimension” in the classical geometric sense in which lines have dimension one, planes have dimension two, etc.

A *quadric* in  $\text{PG}(N, q)$ ,  $N \geq 1$ , is a hypersurface of degree 2 which consists of all points, the coordinates of which satisfy a quadratic equation of the form  $\sum_{i,j=0;i \leq j}^N a_{ij} X_i X_j = 0$ , with  $a_{ij} \in GF(q)$  not all equal to 0. A quadric of  $\text{PG}(N, q)$  is called *singular* if there exists a change of coordinate system which reduces its equation to one in less than  $N + 1$  variables. If  $N = 2n + 2$ ,  $n \geq 0$ , it is known that every non-singular quadric is a parabolic quadric, i.e., for every non-singular quadric of  $\text{PG}(2n + 2, q)$ , there exists a change of coordinate system which reduces its equation to  $X_0^2 + X_1 X_2 + \dots + X_{2n+1} X_{2n+2} = 0$ . Hence, up to change of coordinate system, there is only one non-singular parabolic quadric in  $\text{PG}(2n + 2, q)$ . It is well known that the non-singular parabolic quadric in  $\text{PG}(2n + 2, q)$  contains both points and subspaces of higher dimension when  $n \geq 1$ . A subspace of maximal dimension contained in the quadric is called a *generator*. For the non-singular parabolic quadric  $Q(2n + 2, q)$ , the generators are  $n$ -dimensional subspaces. For detailed information about quadrics in projective spaces, we refer to [13].

A *polarity*  $\alpha$  of  $\text{PG}(N, q)$ ,  $N \geq 2$ , is an involutory permutation of the set of all subspaces of  $\text{PG}(N, q)$  which reverses inclusion between subspaces of  $\text{PG}(N, q)$ ; i.e.,  $\pi_i \subset \pi_j$  if and only if  $\pi_j^\alpha \subset \pi_i^\alpha$ . Every polarity is induced by a semi-linear mapping of the underlying vector space onto its dual, and hence a matrix  $A$  and field automorphism  $\theta$  can be associated with it. We will restrict to *non-degenerate* polarities and hence to non-singular matrices  $A$ . We call the polarity *symplectic* if the transposed matrix  $A^t = -A$  and the field automorphism  $\theta$  is the identity. If  $q = 2^h$ , we add the extra condition that all diagonal elements of  $A$  equal 0. Necessarily,  $N$  must be odd if  $A$  is non-singular. Up to a change in the coordinate system, there is only one symplectic polarity. Consider the projective geometry  $\text{PG}(2n + 1, q)$ . The *symplectic space*  $W(2n + 1, q)$  is the set of all points of  $\text{PG}(2n + 1, q)$  together with all *isotropic subspaces*  $\pi$  of  $\text{PG}(2n + 1, q)$ , i.e., subspaces such that  $\pi^\alpha \subseteq \pi$ , with  $\alpha$  the symplectic polarity of  $\text{PG}(2n + 1, q)$ . It is well known that  $n$  is the maximal dimension of an isotropic subspace. We call a maximal isotropic subspace a *generator*. For detailed information about polarities of projective spaces, we refer to [12].

Both the quadric  $Q(2n + 2, q)$ ,  $q$  odd, and the symplectic space are examples of *classical polar spaces*. A complete description of classical polar spaces can be found in [13]. In this work, classical polar spaces are described as in our introduction, using quadrics, Hermitian varieties and symplectic spaces. An alternative, more unified description using sesquilinear forms of vector spaces can be found in [17], although this work is more an overview comparing the two approaches with the axiomatic description of polar spaces which can be found in [18]. A survey paper about the subject and the different ways of describing polar spaces is [5].

The following definitions will be given for finite classical polar spaces, but we will only use them with the given examples in mind. Consider a classical polar space  $\mathcal{P}$ . An *ovoid*  $\mathcal{O}$  is a set of points of  $\mathcal{P}$  such that every generator of  $\mathcal{P}$  meets  $\mathcal{O}$  in exactly one point. A *blocking set*  $\mathcal{B}$  is a set of points of  $\mathcal{P}$  such that every generator of  $\mathcal{P}$  meets  $\mathcal{B}$  in at least one point. A *spread*  $\mathcal{S}$  is a set of generators of  $\mathcal{P}$  partitioning the point set of  $\mathcal{P}$ . A *cover*  $\mathcal{C}$  is a set of generators of  $\mathcal{P}$  such that every point of  $\mathcal{P}$  lies on at least one generator of  $\mathcal{C}$ . A survey paper on ovoids and spreads of finite classical polar spaces is [19].

For the non-singular parabolic quadric  $Q(2n + 2, q)$  in  $\text{PG}(2n + 2, q)$ , the size of an ovoid is  $q^{n+1} + 1$  and the size of a blocking set is  $q^{n+1} + 1 + r$ ,  $r \geq 0$ . For the symplectic space  $W(2n + 1, q)$ , the size of an ovoid is again  $q^{n+1} + 1$  and the size of a blocking set is again  $q^{n+1} + 1 + r$ ,  $r \geq 0$ . For  $q$  even, there is an important relation between these two polar spaces. Projecting all points and subspaces of  $Q(2n + 2, q)$ ,  $q$  even, from the nucleus of this quadric onto a hyperplane  $\text{PG}(2n + 1, q)$  yields the space  $W(2n + 1, q)$

in  $\text{PG}(2n + 1, q)$ . Hence, a minimal blocking set of  $\text{Q}(2n + 2, q)$  corresponds to a minimal blocking set of  $\text{W}(2n + 1, q)$ , if  $q$  is even.

For the symplectic space  $\text{W}(3, q)$ , the size of a spread is  $q^2 + 1$ , while the size of a cover is  $q^2 + 1 + r$ ,  $r \geq 0$ , and the generators are lines. It is known that  $\text{W}(3, q) \cong \text{Q}(4, q)^D$  [18];  $\text{Q}(4, q)^D$  denotes the *dual* of  $\text{Q}(4, q)$  which is the structure obtained from  $\text{Q}(4, q)$  by interchanging the role of points and lines. Hence, a blocking set of  $\text{Q}(4, q)$  corresponds to a cover of  $\text{W}(3, q)$ , and vice versa. Furthermore,  $\text{Q}(4, q)$  is self dual if and only if  $q$  is even, and so  $\text{Q}(4, q) \cong \text{W}(3, q)$  if and only if  $q$  is even.

A blocking set  $\mathcal{B}$  is called *minimal* if  $\mathcal{B} \setminus \{p\}$  is not a blocking set for every  $p \in \mathcal{B}$ . This is equivalent to the following property: on every point  $p \in \mathcal{B}$ , there is at least one generator meeting  $\mathcal{B}$  only in  $p$ . A cover  $\mathcal{C}$  is called *minimal* if  $\mathcal{C} \setminus \{L\}$  is not a cover for every line  $L \in \mathcal{C}$ . Using the above isomorphism, a minimal blocking set of  $\text{Q}(4, q)$  corresponds to a minimal cover of  $\text{W}(3, q)$ , and vice versa.

Consider now a minimal blocking set  $\mathcal{K}$  of the parabolic quadric  $\text{Q}(2n + 2, q)$ . The following lemma describes a relation between blocking sets of  $\text{Q}(2n + 2, q)$  and blocking sets of  $\text{Q}(2n, q)$ .

**Lemma 1** *Consider a point  $p \in \text{Q}(2n + 2, q) \setminus \mathcal{K}$ . If  $\mathcal{K}_p$  is the projection of  $\mathcal{K} \cap T_p(\text{Q}(2n + 2, q))$  onto  $\text{Q}(2n, q)$ , the base of the tangent cone  $T_p(\text{Q}(2n + 2, q)) \cap \text{Q}(2n + 2, q)$ , then  $\mathcal{K}_p$  is a minimal blocking set of  $\text{Q}(2n, q)$  if  $|\mathcal{K}| \leq q^{n+1} + q^{n-1}$ .*

**Proof.** We refer to [8] for a proof of this lemma. □

Using information about minimal blocking sets of  $\text{Q}(4, q)$ , different from an ovoid, together with other techniques, De Beule and Storme [8] were able to characterize the smallest minimal blocking sets of  $\text{Q}(6, q)$ ,  $q$  even and  $q \geq 32$ . They relied on the following theorem from [10] providing information about blocking sets of  $\text{Q}(4, q)$ ,  $q$  even.

**Theorem 1** *A blocking set of  $\text{Q}(4, q)$ , for  $q$  even,  $q \geq 32$ , with size  $q^2 + 1 + r$  and  $0 < r \leq \sqrt{q}$ , consists of an ovoid and  $r$  extra points.*

Using the symplectic space  $\text{W}(2n + 1, q)$ , K. Metsch proved independently the characterization of the smallest minimal blocking sets of  $\text{Q}(2n + 2, q)$ ,  $q$  even [15]. But the techniques he used do not give results for  $\text{Q}(2n + 2, q)$ ,  $q$  odd. In [8], we managed to prove the same characterization of the smallest minimal blocking sets of  $\text{Q}(2n + 2, q)$ ,  $q = 5, 7$ ,  $n \geq 2$ , using an extension of Theorem 1 for these values of  $q$ .

The similar result of Theorem 1 for  $q$  odd is an open problem. For  $q = 3$ , one can prove an analogous result [7]. For  $q = 5, 7$ , we performed a computer search to exclude the existence of a minimal blocking set of  $\text{Q}(4, q)$  of size  $q^2 + 2$ , and of a minimal blocking set of  $\text{Q}(4, q = 7)$  of size  $q^2 + 3$  satisfying a special property. This is sufficient to prove the characterization theorems in [8].

We obtained the following theorem. We note that a *multiple line* of a blocking set  $\mathcal{B}$  of  $\text{Q}(4, q)$  is a line of  $\text{Q}(4, q)$  containing at least two points of  $\mathcal{B}$ .

**Lemma 2** *A minimal blocking set of  $\text{Q}(4, q)$ ,  $q = 5, 7$ , different from an ovoid, has at least  $q^2 + 3$  points. A minimal blocking set of  $\text{Q}(4, 7)$ , with the property that all multiple lines are blocked by exactly 3 points, has at least  $q^2 + 4$  points.*

## 2 Mathematical setup

Our first aim is to prove that a minimal blocking set of  $\text{Q}(4, q)$ ,  $q = 5, 7$ , different from an ovoid, has size at least  $q^2 + 3$ . Therefore we will exclude the existence of a minimal blocking set of size  $q^2 + 2$ . We need the following definitions.

Consider a minimal cover  $\mathcal{C}$  of  $\text{W}(3, q)$ . The *excess* of  $\mathcal{C}$  is the number of lines of  $\mathcal{C}$  minus  $q^2 + 1$ . A point is called a *multiple point* or an *excess point* when it lies on at least two lines of  $\mathcal{C}$ . The *excess* of a

point is the number of lines of the cover passing through this point, minus one. The *weight of a line with respect to a given cover* is the minimum of the excesses of the points belonging to this line.

A *blocking set*  $\mathcal{B}$  of the projective plane  $\text{PG}(2, q)$  is a set of points such that each line of  $\text{PG}(2, q)$  contains at least one point of  $\mathcal{B}$ . A blocking set  $\mathcal{B}$  containing a line of  $\text{PG}(2, q)$  is called a *trivial* blocking set. When  $\mathcal{B}$  does not contain a line, then  $\mathcal{B}$  is called a *non-trivial* blocking set. Presently, it is known that when  $q$  is a square, the smallest non-trivial blocking sets of  $\text{PG}(2, q)$  have size  $q + \sqrt{q} + 1$  and are Baer subplanes [4]. When  $q$  is prime,  $q > 2$ , then the smallest non-trivial blocking sets have size  $\frac{3(q+1)}{2}$  [2], and when  $q$  is non-square,  $q = p^h$ ,  $h > 2$ ,  $p$  prime, with  $c_2 = c_3 = 2^{-1/3}$  and  $c_p = 1$  for  $p > 3$ , then the smallest non-trivial blocking sets have size at least  $q + c_p q^{2/3} + 1$  [3].

Let  $\mathcal{L}$  be a collection of lines of  $\text{PG}(3, q)$ , where each line is accorded a positive integer, called its *weight*. The set of points which lie on at least one element of  $\mathcal{L}$ , is called the *sum* of the lines  $\mathcal{L}$ . Furthermore, the *weight* of a point in the sum of lines  $\mathcal{L}$  is the sum of the weights of the lines of  $\mathcal{L}$  passing through  $p$ .

A *classical generalized quadrangle* is a rank 2 example of a classical polar space, i.e., a classical polar space containing points and lines, but no higher dimensional spaces. If  $s + 1$  is the number of points on a line and  $t + 1$  is the number of lines through a point, the generalized quadrangle has *order*  $(s, t)$  and is denoted by  $\mathcal{GQ}(s, t)$ . The quadric  $\text{Q}(4, q)$  and the symplectic space  $\text{W}(3, q)$  are examples of classical generalized quadrangles of order  $(q, q)$ . Classical generalized quadrangles form a subset of the set of all generalized quadrangles, which are intensively studied in [18]. If  $\mathcal{S}$  is a spread of a generalized quadrangle of order  $(s, t)$ , then its size is necessarily  $st + 1$ .

From [10], we have the following theorem.

**Theorem 2** *Let  $\mathcal{C}$  be a cover of a classical generalized quadrangle  $\mathcal{S}$  of order  $(q, t)$  embedded in  $\text{PG}(d, q)$ . Let  $|\mathcal{C}| = qt + 1 + r$ , with  $q + r$  smaller than the cardinality of the smallest non-trivial blocking sets in  $\text{PG}(2, q)$ . Then the multiple points of  $\mathcal{C}$  form a sum of lines of  $\text{PG}(d, q)$ , where the weight of a line in this sum is equal to the weight of this line with respect to the cover, and with the sum of the weights of the lines equal to  $r$ .*

Using this theorem and the fact that  $\text{W}(3, q) \cong \text{Q}(4, q)^D$  (see [18]), we obtain important structural information about minimal blocking sets of  $\text{Q}(4, q)$  of small size. For example, suppose that  $\mathcal{B}$  is a minimal blocking set of  $\text{Q}(4, q)$ ,  $q > 2$ , of size  $q^2 + 2$ . This corresponds to a minimal cover of  $\text{W}(3, q)$  of size  $q^2 + 2$ . Since  $q + 2$  is smaller than the cardinality of the smallest non-trivial blocking sets of  $\text{PG}(2, q)$ , we can apply Theorem 2, hence all multiple points of the cover are the points of one line of  $\text{PG}(3, q)$ , not belonging to the cover since the cover is minimal. Moreover, it is also possible to prove that this latter line is a line of  $\text{W}(3, q)$ . Translated to the dual situation, we obtain that there exists a point  $p_1 \in \text{Q}(4, q)$  on which there are  $q + 1$  generators of  $\text{Q}(4, q)$ , all containing exactly two points of  $\mathcal{B}$ .

To prove the non-existence of such a blocking set  $\mathcal{B}$ , we can proceed with the following steps. We note that a *partial blocking set* is a set of points of  $\text{Q}(4, q)$  which possibly can be extended to a blocking set of size  $q^2 + 2$ ,  $q = 5, 7$ , or size  $q^2 + 3$ ,  $q = 7$ .

- Step 1. Select the special point  $p_1$  of  $\text{Q}(4, q)$  on which the multiple lines pass. Since the stabilizer group  $\text{PGO}(5, q)$  of  $\text{Q}(4, q)$  acts transitively on the set of points of  $\text{Q}(4, q)$ , we can select this point  $p_1$  arbitrarily.
- Step 2. Now select some points  $p_2, \dots, p_k$  which must belong to the blocking set, and which do not lie on multiple lines; hence those points are,  $p_1$  included, mutually non collinear on the quadric. We will now use the stabilizer group  $\text{PGO}(5, q)$  of  $\text{Q}(4, q)$  to reduce the number of possibilities. Let  $H$  be the stabilizer group of  $p_1$  on  $\text{Q}(4, q)$ . Stabilizing the set  $\{p_2, \dots, p_k\}$ , we have  $n$  choices for  $p_{k+1}$

if  $n$  is the number of orbits of  $\text{Stab}_{\{p_2, \dots, p_k\}}(H)$  on the set of points of  $Q(4, q)$  not collinear with  $p_1, \dots, p_k$ .

Step 3. Selecting  $p_2, \dots, p_k$  decreases the number of possible points for the blocking set  $\mathcal{B}$  on the  $q + 1$  generators through  $p_1$ , since every  $p_i$ ,  $i = 2, \dots, k$ , is collinear with exactly one point of every generator on  $p_1$ , and no point of  $\mathcal{B}$  on a generator through  $p_1$  can be collinear with a point of  $\mathcal{B}$ , not collinear with  $p_1$ , since the multiple lines of  $\mathcal{B}$  are the generators of  $Q(4, q)$  through  $p_1$ . On each generator  $L_i$ ,  $i = 1, \dots, q + 1$ , through  $p_1$ , we need exactly two points. Hence, if there are still  $n_i$  points on  $L_i$  that can possibly belong to  $\mathcal{B}$ , then we have  $\frac{n_i(n_i-1)}{2}$  possibilities for  $L_i \cap \mathcal{B}$ , and  $\prod_{i=1}^{q+1} \frac{n_i(n_i-1)}{2}$  possibilities in total to extend the partial blocking set of Step 2 to a larger (partial) blocking set, by using points of  $Q(4, q) \setminus \{p_1\}$  collinear with  $p_1$ . If  $n_i < 2$  for some  $i$ , then the configuration  $\{p_2, \dots, p_k\}$  can be excluded. Hence, we find the possible partial blocking sets  $\{p_2, \dots, p_k, r_1, \dots, r_{2(q+1)}\}$ , with  $r_1, \dots, r_{2(q+1)}$  points of  $Q(4, q)$  collinear with  $p_1$ , which might be extendable by points not collinear with  $\{p_1, \dots, p_k, r_1, \dots, r_{2(q+1)}\}$  to a larger (partial) blocking set.

Step 4. Extend the partial blocking sets  $\{p_2, \dots, p_k, r_1, \dots, r_{2(q+1)}\}$  but delete the configuration  $\{p_2, \dots, p_k, r_1, \dots, r_{2(q+1)}\}$  if  $k - 1 + 2(q + 1) + N < q^2 + 2$ , with  $N$  the number of remaining candidate points of  $Q(4, q)$  that can extend this partial blocking set. Do this extension using ordinary backtrack; if you add points  $s_1, \dots, s_i$ ,  $i \geq 1$ , then check the condition  $k - 1 + 2(q + 1) + i + N_i \geq q^2 + 2$  again, where  $N_i$  is the number of points of  $Q(4, q)$  that can still extend the already obtained set to a larger (partial) blocking set, and stop the branch if this condition is not satisfied any more.

Using this approach, we found that for both  $q = 5, 7$ , a minimal blocking set of  $Q(4, q)$  of size  $q^2 + 2$  does not exist. Using the same ideas, if  $q = 7$ , we could also exclude the existence of a minimal blocking set of size  $q^2 + 3$ , having the special property that all multiple lines pass through one point. For, adapting two parameters in Step 2 and Step 3, we can use the same program. In Step 2, we select again a partial blocking set  $\{p_2, \dots, p_k\}$ , but we need now 3 points on every generator on  $p_1$ . Hence, we have to check if  $n_i \geq 3$  for every  $L_i$ . After Step 3, every partial blocking set is a set  $\{p_2, \dots, p_k, r_1, \dots, r_{3(q+1)}\}$ . Finally, we have to check the condition  $k - 1 + 3(q + 1) + N \geq q^2 + 3$  before trying to extend the partial blocking set. Using our program, we could indeed exclude the existence of such a minimal blocking set of  $Q(4, 7)$ .

### 3 A computer search

#### The environment

We use GAP (Groups, Algorithms and Programming) (see [11] for the most recent version and technical information) on a MOSIX cluster of 15 Linux computers. MOSIX is a software package that was specifically designed to enhance the Linux kernel with cluster computing capabilities. The core of MOSIX are adaptive (on-line) load-balancing, memory ushering and file I/O optimization algorithms that respond to variations in the use of the cluster resources, e.g. uneven load distribution or excessive disc swapping due to lack of free memory in one of the nodes. In such cases, MOSIX initiates process migration from one node to another, to balance the load, or to move a process to a node that has sufficient free memory or to reduce the number of remote file I/O operations [16].

There are basically two reasons that explain our choice for GAP. Firstly, it provides a wide range of functions dealing with computations in finite fields and group theory. Together with the built-in functions for vectors and lists, they form a good starting point for dealing with projective geometry over finite fields. Secondly, GAP is a high level, typed, and in some sense object-oriented language. Furthermore, it allows user extendable packages that integrate naturally in the system.

To handle problems in projective geometry over finite fields, a software package called “pg” was developed in two master thesis projects [6]. Together with the above-mentioned functions, it is possible to setup the problem with few commands. For example, with three commands, one can compute all points of an arbitrary quadric in coordinate representation. Furthermore, “pg” contains functions to combine geometrical information with group theoretical functions of GAP. The use of this group theoretical information leads to faster computations. For more information about all built-in GAP functions, we refer to [11]. For more information about “pg”, its possibilities and the manual, we refer to [6].

We will now discuss the GAP code that was used.

### 3.1 Initializations

The following lines of code are to be read as initialization. You will see the command `q=5;`. Of course, this line has to be adapted when using the code for the  $q = 7$  case.

```
RequirePackage("pg");
q := 5;
P := ProjectiveGeometry(4,q);
Q := StandardParabolicQuadric(P);
G := StabilizerGroup(Q);
H := AsPermutationGroup(G);
```

These few lines set up the quadric  $Q(4, q)$  completely. We will also store explicitly the points of the quadric in two representations: as coordinates in `points` and as a natural number in `pointsn`. More details can be found in the manual of the package “pg”.

```
points := ProjectivePoints(Q);
pointsn := Set(PointToNumber(points));
```

Since the stabilizer group of  $Q(4, q)$  acts transitively on its generators (see [12]), it suffices to compute one generator. All the others are found as the orbit of the first one under the action of the stabilizer group  $PGO(5, q)$  on the generators. Using “pg”, we translate the problem into a problem of computing the orbit of a set under the action of a permutation group. GAP commands can do the orbit calculation. We represent a generator as a set of points represented by their natural numbers.

```
p1 := points[1];
tangentpoints := Intersection(ProjectivePoints(TangentHyperplane(Q,p1)),points);
nonsingular := Difference(tangentpoints,[p1]);
p2 := nonsingular[1];
line := SubspaceOfPG([p1,p2]);
linen := Set(PointToNumber(ProjectivePoints(line)));
generatorsn := Orbits(H,[linen],OnSets)[1];
generators := [];
for i in [1..Length(generatorsn)] do
dummy := SubspaceOfPG([NumberToPoint(P,generatorsn[i][1]),
NumberToPoint(P,generatorsn[i][2])]);
Add(generators,dummy);
od;
```

We will frequently use the generators of  $Q(4, q)$  on a given point and the points of  $Q(4, q)$  collinear with a given point of  $Q(4, q)$ . To omit repetitions of calculations, we compute two arrays containing this information.

```

genonp := [];
for i in pointsn do
  dummy := [];
  for x in generatorsn do
    if i in x then
      Add(dummy,x);
    fi;
  od;
  genonp[i] := dummy;
od;
collwithp := [];
for i in pointsn do
  dummy := [];
  for j in genonp[i] do
    dummy := Union(dummy,j);
  od;
  dummy := Set(dummy);
  collwithp[i] := dummy;
od;

```

### 3.2 Step 1

The point  $p_1$  defined above will be the special point on which the multiple lines pass. With the following code, we also store in  $H$  the stabilizer group in  $\text{PGO}(5, q)$  of the point  $p_1$ . The point  $p_2$  of  $\text{Q}(4, q)$ , not collinear with  $p_1$ , will be the first point of the blocking set. We can select  $p_2$  again arbitrarily since the stabilizer group  $H$  of  $p_1$  acts still transitively on the points of  $\text{Q}(4, q)$  not collinear with  $p_1$ .

```

p1 := pointsn[1];
pointsn := Difference(pointsn,collwithp[p1]);
cone := Difference(collwithp[p1],[p1]);
H := Stabilizer(H,p1);
biglist := [];
keuzes := [];
p2 := pointsn[1];

```

At this point, all initializations are done and Step 1 is executed.

### 3.3 Step 2

The function `setup` computes the possible starting configurations  $\{p_2, \dots, p_k\}$  for the blocking set, as explained in Step 2. We can adapt the parameters `param` and `sizecone`. The first parameter is the number  $k - 1$  of points our starting configuration will have. The second one is the minimum total number of possible candidates to extend the partial blocking set  $\{p_2, \dots, p_k\}$  which we need on the  $q + 1$  generators on the point  $p_1$ .

```

setup := function(pointsn,keuze,H,keuzes,cone,param,sizecone,biglist)
  local npointsn,nH,nkeuzes,orbits,orbit,ncone;
  npointsn := Difference(pointsn,collwithp[keuze]);
  nkeuzes := Union(keuzes,[keuze]);
  nH := Stabilizer(H,nkeuzes,OnSets);
  orbits := Orbits(nH,npointsn);

```

```

ncone := Difference(cone,collwithp[keuze]);
if Length(nkeuzes) = param then
  if Length(ncone) >= sizecone then
    Add(biglist,rec(bs := Set(nkeuzes),cone := ncone,length :=Length(ncone)));
  fi;
else
  for orbit in orbits do
    setup(npointsn,orbit[1],H,nkeuzes,ncone,param,sizecone,biglist);
  od;
fi;
end;

```

### 3.4 Step 3: checking the configurations from Step 2

The points of  $Q(4, q) \setminus \{p_1\}$  collinear with  $p_1$  and not collinear with the points from a given partial blocking set  $\{p_2, \dots, p_k\}$  of Step 2, are in the parameter `cone` of `analyselines`. This function `analyselines` checks if there are sufficient points (the parameter `length`) on every generator on  $p_1$  to extend the given partial blocking set with. The parameter `point` is the point  $p_1$ . This function is called by the function `analyse`.

```

analyselines := function(cone,point,length)
local bool,lines,i,l;
bool := true;
lines := genonp[point];
l := Length(lines);
i := 1;
while (bool and (i<=l)) do
  if (Length(Intersection(cone,lines[i])) < length) then
    bool := false;
  fi;
  i := i+1;
od;
return bool;
end;

```

After computing starting configurations with the function `setup`, we have to analyze them. Given the list `biglist` with starting configurations computed with `setup`, we check that:

1. in total, are there sufficient points on the generators on the point  $p_1$  that can extend the starting configuration? If so:
2. can we select on every such generator at least the desired number of points?

The variable `biglist` is a list with records containing all required information. Such a record has a component `bs` containing the partial blocking set  $\{p_2, \dots, p_k\}$ , `cone` containing the points collinear with  $p_1$  which are not excluded by the choice of the points in `bs`, `point` is the point  $p_1$ , `sizecone` is the total number of points of the blocking set on the multiple lines, and `length` is the number of points of the blocking set on a multiple line.

```

analyse := function(biglist,point,sizecone,length)
local dummylist,resultlist;
dummylist := [];

```

```

resultlist := [];
for x in biglist do
  if ((Length(x.cone) >= sizecone) and analyselines(x.cone,point,length)) then
    if not (x.bs in dummylist) then
      Add(resultlist,x);
      Add(dummylist,x.bs);
    fi;
  fi;
od;
return resultlist;
end;

```

The last function computes the number of configurations described in Step 3, following from a given starting configuration described in Step 2.

```

countbranches := function(biglist,point,length)
local x,sum,lines,line,i,mult;
sum := 0;
for x in biglist do
  lines := genonp[point];
  mult := 1;
  for i in [1..Length(lines)] do
    line := Intersection(lines[i],x.cone);
    mult := mult*NrCombinations(line,length);
  od;
  sum := sum+mult;
od;
return sum;
end;

```

### 3.5 Step 3: creating new starting configurations

After the checks of Step 3, we can go further with the analyzed starting configurations  $\{p_2, \dots, p_k\}$  and compute a set of new starting configurations as described in Step 3. We will now discuss the code to execute this step. The first function is of use in the big function `createstartconfs`.

```

analysepbs := function(pbs,point,point)
local list,p,npointsn;
list := [];
npointsn := Difference(ShallowCopy(point),collwithp[point]);
for p in pbs do
  npointsn := Difference(npointsn,collwithp[p]);
od;
return npointsn;
end;

```

The next function is the core of Step 3. The function itself is recursive. Given a partial blocking set `bs`, this function will try to extend it, and, if successful and if the condition on the number of remaining candidates is satisfied, the function calls itself with the freshly extended partial blocking set as variable. We will briefly describe the main ideas behind the function. If we start from a partial blocking set  $\{p_2, \dots, p_k\}$

as described in Step 2, after analyzing such a configuration, we have to extend it with points collinear with  $p_1$ . On each generator on  $p_1$ , we have to select the desired number of points. The parameter **max** will be the number of generators on  $p_1$ , while **step** will be the level of recursion. If **step** > **max**, then we have made a selection of points for all generators on  $p_1$ , and we obtain a partial blocking set  $\{p_2, \dots, p_k, r_1, \dots, r_{2(q+1)}\}$  or  $\{p_2, \dots, p_k, r_1, \dots, r_{3(q+1)}\}$  as described in Step 3. If the number of remaining candidates that can extend this partial blocking set is sufficient to start Step 4, then we store the partial blocking set in a global variable **resultlist**. This approach reduces the number of lists that have to be stored in the memory. We mention briefly the meaning of the other variables: **point** is the point  $p_1$ , **lines** are the generators on  $p_1$ , **pointsn** is the list with the points of  $Q(4, q) \setminus p_1^\perp$ , where  $p_1^\perp$  defines the set of points of  $Q(4, q)$  collinear with  $p_1$ , described via their corresponding natural numbers, **length** the required number of points of the blocking set on the multiple lines, **bs** the partial blocking set, **cone** the set of points different from  $p_1$  and on the generators through  $p_1$ , not collinear with a point of the initial partial blocking set and **sizebs** the size of the blocking set we are looking for.

```

createstartconfs :=
function(point,lines,pointsn,step,max,length,bs,cone,sizebs,resultlist)
local nbs,comb,c,npointsn;
comb := Combinations(Intersection(cone,lines[step]),length);
step := step + 1;
for c in comb do
  if step > max then
    nbs := Union(bs,c);
    npointsn := analysepbs(nbs,pointsn,point);
    if Length(nbs)+Length(npointsn) >= sizebs then
      Add(resultlist,rec(bs := nbs,pointsn := npointsn));
    fi;
  else
    nbs := Union(bs,c);
createstartconfs(point,lines,pointsn,step,max,length,nbs,cone,sizebs,resultlist);
  fi;
od;
end;

```

### 3.6 Step 4

The last function we need is a function which extends a partial blocking set as found in Step 3 to a desired blocking set (if possible), using an ordinary backtrack. If a blocking set of the desired size is found, it is stored in the global variable **resultlist**.

```

createbs := function(pbs,pointsn,sizebs,choices,resultlist)
local npbs,npointsn,list,nchoices,x;
if Length(pbs) = sizebs then
  Add(resultlist,pbs);
else
  list := Difference(pointsn,choices);
  for x in list do
    npbs := Union(pbs,[x]);
    npointsn := Difference(pointsn,collwithp[x]);
    nchoices := Union(choices,[x]);
    if Length(npbs)+Length(npointsn) >= sizebs then

```

```

        createbs(npbs,npointsn,sizesbs,nchoices,resultlist);
    fi;
od;
fi;
end;

```

## 4 The execution of the code and the results

In this section, we describe how the above discussed code is executed and what the results are. The code from Sections 3.1 to 3.4 is in the file `functions2`. After starting GAP, this file is read. Then we can execute Step 1 and Step 2 with two commands. We added the `time` command to give an idea of the computation time. This command gives the computation time in milliseconds. All computations were done on Intel compatible systems, using AMD athlon 750 Mhz single processor machines.

**The case  $q = 5$ .**

```

gap> Read("./q45new/functions2");
gap> time;
26170
gap> setup(pointsn,p2,H,keuzes,cone,3,12,biglist);
gap> time;
10000
gap> Length(biglist);
22
gap> result := analyse(biglist,p1,12,2);
gap> Length(result);
21

```

At this point, we have executed Step 1 and Step 2 for the case  $q = 5$ . It is clear that we go to the instructions of Section 3.5 with 21 starting configurations  $\{p_2, p_3, p_4\}$ . Going into that section, we execute the following commands. We wrote the code of Section 3.5 in the file `functions3`.

```

gap> Read("./q45new/functions3");
gap> resultlist := [];
[ ]
gap> for param in [1..Length(result)] do
> x := result[param];
> sizesbs := q^2+2;
> lines := genonp[p1];
> length := 2;
> createstartconfs(p1,lines,pointsn,1,q+1,length,x.bs,x.cone,sizesbs,resultlist);
> t := time;
> n := countbranches([x],p1,length);
> Print("number: ",param," \n");
> Print("length of resultlist: ",Length(resultlist)," \n");
> Print("checked ",n," configurations\n");

> od;

```

We present the results in the following table.

number	length of resultlist	checked configurations
1	0	162
2	0	81
3	0	36
4	0	162
5	0	9
6	0	81
7	0	9
8	0	9
9	0	36
10	0	81
11	0	729
12	0	81
13	0	9
14	0	81
15	0	81
16	0	9
17	0	9
18	0	81
19	0	1
20	0	1
21	0	1

```
gap> time;
810
gap> Length(resultlist);
0
```

Since the length of `resultlist` is 0, we do not have to execute Step 4. We conclude that a minimal blocking set of  $Q(4,5)$  of size  $q^2 + 2 = 27$  does not exist.

### The case $q = 7$ .

For  $q = 7$ , we will handle two cases: minimal blocking sets of size  $q^2 + 2$  with multiple lines that are blocked by two points, and minimal blocking sets of size  $q^2 + 3$  with multiple lines that are blocked by three points. We will execute Step 1 for the two cases. In both cases, we look for 4 points  $p_2, \dots, p_5$ . In the first case, multiple lines are blocked by two points of the minimal blocking set, so we need at least 16 points collinear with  $p_1$  which are not excluded by the choice of the first 4 points  $p_2, \dots, p_5$ . In the second case, multiple lines are blocked by three points of the minimal blocking set, so we need at least 24 points collinear with  $p_1$  which are not excluded by the choice of the first 4 points  $p_2, \dots, p_5$ . These parameters can be found back in the following code.

```
gap> Read("./q47new/functions2");
gap> time;
68650
gap> setup(pointsn,p2,H,keuzes,cone,4,16,biglist);
gap> time;
2324100
gap> result := analyse(biglist,p1,16,2);
gap> Length(result);
```

3508

```

gap> Read("./q47new/functions2");
gap> time;
74270
gap> setup(pointsn,p2,H,keuzes,cone,4,24,biglist);
gap> time;
2426990
gap> Length(biglist);
3932
gap> result := analyse(biglist,p1,24,3);
gap> Length(result);
3508

```

In both cases, there are 3508 starting configurations from Step 2. Each one of them will give a new set of starting configurations after Step 3. Like in the  $q = 5$  case, some of these can be thrown away; for others, we will have to execute Step 4. Since we have a lot of configurations that can be handled independently from each other, this is the point to parallelize the search. We achieved this by saving the workspace. To execute the next step, we load the workspace and consider a desired starting configuration  $\{p_2, \dots, p_5\}$  on which we execute the instructions of Section 3.5. We write the output in different files. If a starting configuration must be further investigated after Step 3, we save the new workspace. In the file `functions3`, we write the second part of the code. In the file `start`, we write the following GAP commands.

```

Read("./q47new/functions3");
x := result[param];
resultlist := [];
sizebs := q^2+2;
lines := genonp[p1];
length := 2;
createstartconfs(p1,lines,pointsn,1,q+1,length,x.bs,x.cone,sizebs,resultlist);
t := time;
str := "./q47new/results/workspaces/Wq47.create.";
out := OutputTextString(str,true);
PrintTo(out,String(param));
CloseStream(out);
n := countbranches([result[param]],p1,length);
Print("number: ",param," \n");
Print("length of resultlist: ",Length(resultlist)," \n");
Print("checked ",n," configurations\n");
Print("calculating time for confs: ",t,"\n");
if (Length(resultlist) = 0) then
  Print("quitting without saving workspace\n");
  quit;
fi;
Print("Saving Workspace\n");
SaveWorkspace(str);

```

A GAP job can be started with the following command. In this command, we write the explicit value for the parameter `param`. This is why the for loop, present in the code for  $q = 5$ , does not appear here.

```
(echo "param := 1;" ; cat ./q47new/start ) | gap4r3 -l "./;/opt/gap4r3/" \
-L ./q47new/Wq47.analyse > ./q47new/results/output.1
```

We give two examples of output files, with `param=174` and `param=1782` respectively. The first case corresponds to a starting configuration which cannot be extended to a minimal blocking set of size  $q^2 + 2$ ; the second case to a starting configuration which still possibly can be extended to a minimal blocking set of size  $q^2 + 2$ .

```
GAP4, Version: 4.3fix3 of September 12, 2002, i686-pc-linux-gnu-gcc
Components:  small, small2, small3, small4, small5, small6, small7,
              small8, id2, id3, id4, id5, id6, trans, prim loaded.
Packages:    tomlib, ctbllib, pg loaded.
gap> 174
gap> gap> rec( bs := [ 3, 15, 46, 1398 ],
  cone := [ 56, 74, 188, 224, 255, 463, 485, 525, 654, 662, 670, 848, 887,
           946, 1118, 1191, 1317, 1594, 1686, 1712, 1789, 1874, 1925, 2017, 2050,
           2106, 2209, 2260, 2417, 2519, 2574, 2730 ], length := 32 )
gap> [ ]
gap> 51
gap> [ [ 2, 255, 670, 1103, 1662, 2260, 2478, 2574 ],
      [ 2, 74, 216, 1118, 1191, 1843, 1851, 2417 ],
      [ 2, 5, 946, 1410, 1734, 1789, 2017, 2050 ],
      [ 2, 56, 304, 350, 463, 763, 1594, 2519 ],
      [ 2, 188, 224, 525, 866, 1692, 1925, 2223 ],
      [ 2, 654, 662, 1228, 1614, 1686, 1828, 2730 ],
      [ 2, 485, 1317, 1360, 1560, 1712, 2209, 2327 ],
      [ 2, 4, 353, 848, 887, 1665, 1874, 2106 ] ]
gap> 2
gap> gap> 584460
gap> "./q47new/results/workspaces/Wq47.create."
gap> OutputTextString(40)
gap> gap> gap> 1679616
gap> number: 174
gap> length of resultlist: 0
gap> checked 1679616 configurations
gap> calculating time for confs: 584460
gap> > quitting without saving workspace
```

Now follows the second example with `param=1782`.

```
GAP4, Version: 4.3fix3 of September 12, 2002, i686-pc-linux-gnu-gcc
Components:  small, small2, small3, small4, small5, small6, small7,
              small8, id2, id3, id4, id5, id6, trans, prim loaded.
Packages:    tomlib, ctbllib, pg loaded.
gap> 1782
gap> gap> rec( bs := [ 3, 15, 442, 2132 ],
  cone := [ 56, 74, 188, 255, 463, 485, 525, 654, 662, 670, 848, 887, 946,
           1118, 1191, 1228, 1317, 1410, 1560, 1594, 1712, 1734, 1789, 1843, 1874,
           2017, 2106, 2223, 2260, 2519, 2730 ], length := 31 )
```

```

gap> [ ]
gap> 51
gap> [ [ 2, 255, 670, 1103, 1662, 2260, 2478, 2574 ],
      [ 2, 74, 216, 1118, 1191, 1843, 1851, 2417 ],
      [ 2, 5, 946, 1410, 1734, 1789, 2017, 2050 ],
      [ 2, 56, 304, 350, 463, 763, 1594, 2519 ],
      [ 2, 188, 224, 525, 866, 1692, 1925, 2223 ],
      [ 2, 654, 662, 1228, 1614, 1686, 1828, 2730 ],
      [ 2, 485, 1317, 1360, 1560, 1712, 2209, 2327 ],
      [ 2, 4, 353, 848, 887, 1665, 1874, 2106 ] ]
gap> 2
gap> gap> 237510
gap> "./q47new/results/workspaces/Wq47.create."
gap> OutputTextString(40)
gap> gap> gap> 699840
gap> number: 1782
gap> length of resultlist: 1
gap> checked 699840 configurations
gap> calculating time for confs: 237510
gap> > > gap> Saving Workspace
gap> true

```

In total, 143 workspaces corresponding to starting configurations from Step 3 were saved. We can handle Step 4 similarly. The following GAP code, together with the function `createbs`, is put in the file `start2`.

```

biglist := [];
str := "./q47new/results/workspaces/results/Wq47.createbs.";
out := OutputTextString(str,true);
PrintTo(out,String(param));
CloseStream(out);
Print("Length of resultlist: ",Length(resultlist),"\\n");
for x in resultlist do
  createbs(x.bs,x.pointsn,51,[],biglist);
od;
t := Runtime();
Print("Length of biglist: ",Length(biglist),"\\n");
Print("Calculating time: ",t,"\\n");
if Length(biglist)=0 then
  Print("Quitting without saving workspace\\n");
  quit;
fi;
Print("Saving workspace\\n");
SaveWorkspace(str);

```

The following shell command starts one of the 143 jobs. We give the variable `param` the number of the configuration we want to handle. The parameter `param` takes on 143 values of the integers between 1 and 3508, where 3508 was the number of starting configurations after Step 2. For instance, 53 is one of those values for `param`.

```
(echo "param := 53;" ; cat ./q47new/start2 ) | gap4r3 -l "./;/opt/gap4r3/" -L \
"./q47new/results/workspaces/Wq47.create.53" >
./q47new/results/workspaces/results/output.53
```

No minimal blocking set of size 51 is found in these 143 jobs. All output files look like the following.

```
GAP4, Version: 4.3fix3 of September 12, 2002, i686-pc-linux-gnu-gcc
Components:  small, small2, small3, small4, small5, small6, small7,
              small8, id2, id3, id4, id5, id6, trans, prim loaded.
Packages:    tomlib, ctbllib, pg loaded.
gap> 53
gap> function( pbs, pointsn, sizebs, choices, resultlist ) ... end
gap> gap> [ ]
gap> "./q47new/results/workspaces/results/Wq47.createbs."
gap> OutputTextString(50)
gap> gap> gap> Length of resultlist: 1
gap> > > gap> 12020
gap> Length of biglist: 0
gap> Calculating time: 12020
gap> > Quitting without saving worksapce
```

Executing Step 3 took approximately 827697 seconds; executing Step 4 took 1764 seconds.

Finally, we describe the second case. We have already described the execution of Step 2. For Step 3, we have to execute the same code as in the previous case, but with two parameters adapted. To execute Step 3, we have to change two lines in the file `start`; the line initializing the size of the blocking set and the line initializing the number of points of the blocking set on a multiple line.

```
...
sizebs := q^2+3;
length := 3;
...
```

With the same shell command as in the previous case, we can execute Step 3 for one of the 3508 starting configurations  $\{p_2, \dots, p_5\}$ . All output files look like the following.

```
gap> 1012
gap> gap> rec( bs := [ 3, 15, 136, 534 ],
  cone := [ 74, 188, 216, 224, 255, 304, 463, 485, 525, 654, 662, 848, 887,
           946, 1118, 1191, 1228, 1317, 1560, 1594, 1686, 1734, 1874, 1925, 2017,
           2050, 2106, 2209, 2260, 2417, 2519, 2574, 2730 ], length := 33 )
gap> [ ]
gap> 52
gap> [ [ 2, 255, 670, 1103, 1662, 2260, 2478, 2574 ],
      [ 2, 74, 216, 1118, 1191, 1843, 1851, 2417 ],
      [ 2, 5, 946, 1410, 1734, 1789, 2017, 2050 ],
      [ 2, 56, 304, 350, 463, 763, 1594, 2519 ],
      [ 2, 188, 224, 525, 866, 1692, 1925, 2223 ],
      [ 2, 654, 662, 1228, 1614, 1686, 1828, 2730 ],
      [ 2, 485, 1317, 1360, 1560, 1712, 2209, 2327 ],
```

```
[ 2, 4, 353, 848, 887, 1665, 1874, 2106 ] ]
gap> 3
gap> gap> 30990
gap> "./q47new/results/workspaces/Wq47.create."
gap> OutputTextString(40)
gap> gap> gap> 102400
gap> number: 1012
gap> length of resultlist: 0
gap> checked 102400 configurations
gap> calculating time for confs: 30990
gap> > quitting without saving workspace
```

So the length of `resultlist` is always zero.

To execute Step 3 for all 3508 possibilities took 25272 seconds. As no configuration admits a partial blocking set which possibly can be extended to a minimal blocking set  $\mathcal{B}$  of size  $q^2 + 3$ , with multiple lines containing three points of  $\mathcal{B}$ , Step 4 does not have to be executed.

We conclude that a minimal blocking set  $\mathcal{B}$  of  $Q(4, 7)$  of size 51 does not exist, and that a minimal blocking set  $\mathcal{B}$  of  $Q(4, 7)$  of size 52 with multiple lines blocked by three points of  $\mathcal{B}$ , does not exist.

## 5 Conclusions

We conclude that:

- a minimal blocking set  $\mathcal{B}$  of  $Q(4, 5)$  of size  $q^2 + 2 = 27$  does not exist,
- a minimal blocking set  $\mathcal{B}$  of  $Q(4, 7)$  of size  $q^2 + 2 = 51$  does not exist,
- a minimal blocking set  $\mathcal{B}$  of  $Q(4, 7)$  of size  $q^2 + 3 = 52$ , with the property that all multiple lines are blocked by exactly three points of  $\mathcal{B}$ , does not exist.

These results about minimal blocking sets of  $Q(4, q)$ ,  $q = 5, 7$ , different from an ovoid, have direct applications in [8]. However, it would be interesting to extend this information for larger values of  $q$ , since the classification of the smallest minimal blocking sets of  $Q(4, q)$ ,  $q$  odd, different from an ovoid, is still an important open problem. From this point of view, excluding certain possibilities, as in this paper, or finding examples of a certain size, can guide the theoretical research in the good direction. To achieve this goal, one needs to improve the computational methods used here, since increasing the value of  $q$  will increase the computation time exponentially. On the other hand, with the software we used (GAP and “pg”) on a MOSIX cluster of Linux computers, it is easy to setup the problem and to explore small examples in a reasonable time.

## Acknowledgement

The authors want to thank Lic. Geert Vernaeve for his technical assistance.

## References

- [1] A. Beutelspacher and U. Rosenbaum. *Projective Geometry*. Cambridge University Press, 1998.
- [2] A. Blokhuis. On the size of a blocking set in  $PG(2, p)$ . *Combinatorica*, 14(1):111–114, 1994.

- [3] A. Blokhuis, L. Storme, and T. Szőnyi. Lacunary polynomials, multiple blocking sets and Baer subplanes. *J. London Math. Soc. (2)*, 60(2):321–332, 1999.
- [4] A. A. Bruen. Baer subplanes and blocking sets. *Bull. Amer. Math. Soc.*, 76:342–344, 1970.
- [5] F. Buekenhout and E. E. Shult. On the foundations of polar geometry. *Geom. Dedicata*, 3:155–170, 1974.
- [6] J. De Beule, P. Govaerts, and L. Storme. *Projective Geometries*, a share package for GAP. <http://cage.ugent.be/~jdebeule/pg>
- [7] J. De Beule and L. Storme. The smallest minimal blocking sets of  $Q(6, q)$ ,  $q$  even. *J. Combin. Des.*, 11(4):290–303, 2003.
- [8] J. De Beule and L. Storme. On the smallest minimal blocking sets of  $Q(2n, q)$ , for  $q$  an odd prime. *Discrete Math., Proceedings of the First Irsee Conference on Finite Geometry (Irsee, 2003)*, to appear.
- [9] J. Dittmann, A. Behr, M. Stabenau, P. Schmitt, J. Schwenk, and J. Ueberberg. *Combining digital Watermarks and collusion secure Fingerprints for digital Images*. <http://www.ipsi.fraunhofer.de/~dittmann/pdf/spie-dittmann.pdf>
- [10] J. Eisfeld, L. Storme, T. Szőnyi, and P. Sziklai. Covers and blocking sets of classical generalized quadrangles. *Discrete Math.*, 238(1-3):35–51, 2001. Proceedings of *The Third International Shanghai Conference on Designs, Codes and Finite Geometries* (Shanghai, China, May 14-18, 1999).
- [11] The GAP Group, *GAP – Groups, Algorithms, and Programming, Version 4.3*; 2002. <http://www.gap-system.org>
- [12] J. W. P. Hirschfeld. *Projective Geometries Over Finite Fields*. The Clarendon Press, Oxford University Press, New York, second edition, 1998.
- [13] J. W. P. Hirschfeld and J. A. Thas. *General Galois Geometries*. Oxford Mathematical Monographs. The Clarendon Press, Oxford University Press, New York, 1991. Oxford Science Publications.
- [14] F. J. MacWilliams and N. J. A. Sloane. *The Theory of Error Correcting Codes*. North-Holland, 1977.
- [15] K. Metsch. Small point sets that meet all generators of  $W(2n+1, q)$ . *Des. Codes Cryptogr.*, to appear.
- [16] MOSIX. <http://www.mosix.org>
- [17] A. Pasini. *Diagram Geometries*. Oxford Science Publications. The Clarendon Press, Oxford University Press, New York, 1994.
- [18] S. E. Payne and J. A. Thas. *Finite Generalized Quadrangles*. Pitman (Advanced Publishing Program), Boston, MA, 1984.
- [19] J. A. Thas. Ovoids and spreads of finite classical polar spaces. *Geom. Dedicata*, 10(1-4):135–143, 1981.
- [20] N. Thomas. Projective geometry. <http://www.anth.org.uk/NCT/>