The size of minimal blocking sets of $Q(4, q)$

J. De Beule, L. Storme, A. Hoogewijs  
Ghent University

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 determine the minimal number of points of a minimal blocking set of $Q(4, q)$, for $q = 5, 7$.

1 Introduction and definitions

Consider the projective geometry $PG(N, q)$. A quadric in $PG(N, q)$, $N \geq 1$ is a hypersurface of degree 2 which consists of all points, the coordinates of which satisfy a quadratic equation of the form

$$\sum_{i,j=0}^{N} a_{ij} X_i X_j = 0,$$

with not all $a_{ij}$ equal to 0. A quadric is called singular if there exists a change of coordinate system which reduces its equation to one in fewer 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 there exists a change of coordinate system which reduces its equation to $X_0^2 + X_1 X_2 + \ldots + X_{2n+1} X_{2n+2} = 0$, hence up to change of coordinate system, there is only one non-singular parabolic quadric in $PG(2n + 2, q)$. It is well known that the non-singular parabolic quadric contains besides points, also subspaces of higher dimension. A subspace of maximum 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 [10].

A polarity $\alpha$ of $PG(N, q)$, $N \geq 2$ is a involutory permutation of the set of all subspaces of $PG(N, q)$ which reverses inclusion between subspaces of $PG(N, q)$; i.e. $\pi_i \subset \pi_j$ if and only if $\pi_j \subset \pi_i$. 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. We call the polarity symplectic if the transposed matrix $A^t = -A$ and the field automorphism is the identity. If $q = 2^h$, we add the extra condition that all diagonal elements equal 0. Necessarily $N$ must be odd if $A$ must be non-singular. Up to change of coordinate system, there is only one symplectic polarity. Consider the projective geometry $PG(2n + 1, q)$. The symplectic space $W(2n + 1, q)$ is the set of all points together with all isotropic subspaces $\pi$ of $PG(2n + 1, q)$, subspaces such that $\pi^\alpha \subset \pi$, with $\alpha$ the symplectic polarity of $PG(2n + 1, q)$. 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 [9].

Both the quadric $Q(2n + 2, q)$ and the symplectic space are examples of classical polar spaces. A complete description of classical polar spaces can be found in [10]. 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 [12], although this work is more an overview comparing the two approaches with the axiomatic description of polar spaces which can be found in [13]. A survey paper about the subject and the different ways of describing polar spaces is [4].
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 such that every generator of $\mathcal{P}$ meets $\mathcal{O}$ in exactly one point. A blocking set $\mathcal{B}$ is a set of points such that every generator of $\mathcal{P}$ meets $\mathcal{B}$ in at least one point. A cover $\mathcal{C}$ is a set of generators such that every point lies on at least one generator of $\mathcal{P}$. A survey paper on ovoids and spreads of finite classical polar spaces is [15].

Consider the non-singular parabolic quadric $Q(2n+2,q)$ in PG$(2n+2,q)$. The size of an ovoid is $q^{n+1}+1$ and the size of a blocking set is $q^{n+1}+1+r$, $r > 0$. Consider 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 $q^{n+1}+1+r$, $r > 0$. For $q$ even there is an important relation between these two polar spaces. Projecting all points and subspaces of $Q(2n+2,q)$ from the nucleus of this quadric yields the space $W(2n+1,q)$ in PG$(2n+1,q)$. Hence a minimal blocking set of $Q(2n+2,q)$ corresponds to a minimal blocking set of $W(2n+1,q)$ if $q$ is even.

Consider the symplectic space $W(3,q)$, the size of a spread is $q^2+1$, while the size of a cover is $q^2+1+r$, $r > 0$ and the generators are lines. It is known that $W(3,q) \cong M_{q^3}(q)$, the dual of $Q(4,q)$. Furthermore $Q(4,q)$ is self-dual if and only if $q$ is even, and so $Q(4,q) \cong W(3,q)$ if and only if $q$ is even, which follows also from the above isomorphism. Hence a blocking set of $Q(4,q)$ corresponds to a cover of $W(3,q)$ and vice versa.

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 $\mathcal{B}$ 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 $Q(4,q)$ corresponds to a minimal cover of $W(3,q)$. In general this isomorphism is only valid if $q$ is even.

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

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

**Proof.** We refer to [5] for a proof of this lemma.

Using information about the minimal size of a minimal blocking set of $Q(4,q)$ and other techniques, we were able to characterize the smallest minimal blocking sets of $Q(6,q)$, $q$ even and $q \geq 32$. Information about blocking sets of $Q(4,q)$, $q$ even is provided by the following theorem from [7].

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

Using the symplectic space $W(2n+1,q)$ K. Metsch proves independently the characterization of the smallest minimal blocking sets of $Q(2n+2,q)$, $q$ even, [11]. But the techniques he used do not give results for $q$ odd in the $Q(2n,q)$. In [6], we managed to prove the same characterization of the smallest minimal blocking sets of $Q(2n,q)$, $q = 3, 5, 7$, using a extension of Theorem 1 for these values of $q$.

The general $q$ odd case seems to be an open problem. For $q = 3$ one can prove the same result. For $q = 5, 7$ we do a computer search to exclude the existence of a minimal blocking set of $Q(4,q)$ of size $q^2 + 2$, and a minimal blocking set of $Q(4,q = 7)$ of size $q^2 + 3$ satisfying a special property. This is sufficient to prove the characterization theorems in [5].

We obtained the following lemma

**Lemma 2** A minimal blocking set of $Q(4,q)$, $q = 5, 7$, has at least $q^2 + 3$ points. A minimal blocking set of $Q(4,7)$ 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 $Q(4, q)$, $q = 5, 7$, 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 $C$ of $W(3, q)$. The excess of $C$ is the number of lines of $C$ minus $q^2 + 1$. A point is called a multiple point or an excess point when it lies on at least two lines of $C$. 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 $B$ of the projective plane $PG(2, q)$ is a set of points such that each line of $PG(2, q)$ contains at least one point of $B$. A blocking set $B$ containing a line of $PG(2, q)$ is called a trivial blocking set. When $B$ does not contain a line, then it is called a non-trivial blocking set. Presently it is known that when $q$ is square, the smallest non-trivial blocking sets of $PG(2, q)$ have size $q + \sqrt{q} + 1$ and are Baer subplanes [3]. When $q$ is prime, $q > 2$, then the smallest non-trivial blocking sets have size $\frac{3(q+1)}{2}$ [1], 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 $q + c_p q^{2/3} + 1$ [2].

Let $L$ be a collection of lines of $PG(3, q)$, where each line is accorded a positive integer, called its weight. The set of points, which lie on at least one element of $L$ is called the sum of the lines $L$. Furthermore, the weight of a point in the sum of lines $L$ is the sum of the weights of the lines of $L$ 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 $s$ is the number of points on a line and $t$ is the number of lines through a point, the generalized quadrangle has order $(s, t)$ and is denoted with $GQ(s, t)$. The quadric $Q(4, q)$ and the symplectic space $W(3, q)$ are examples of classical generalized quadrangles of order $(q, q)$. Classical generalized quadrangles are a subset of generalized quadrangles, which are intensively studied in [13]. If $S$ is a spread of a generalized quadrangle, then its size is necessarily $st + 1$.

From [7] we have the following theorem.

Theorem 2 Let $C$ be a cover of a classical generalized quadrangle $Q = GQ(q, t)$ in $PG(N, q)$. Let $|C| = qt + 1 + r$, with $q + r$ smaller then the cardinality of the smallest non-trivial blocking set in $PG(2, q)$. Then the multiple points form a sum of lines, contained in $Q$, where the weight of a line in this sum is equal to the weight of this lines 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 $W(3, q) \cong Q(4, q)^D$, we obtain important structural information about minimal blocking sets of $Q(4, q)$ of small size. For, suppose $B$ is a minimal blocking set of $Q(4, q)$ of size $q^2 + 2$. This corresponds to a minimal cover of $W(3, q)$ of size $q^2 + 2$. Since $q + 2$ is smaller than the cardinality of the smallest non-trivial blocking sets of $PG(2, q)$, we can apply Theorem 2, hence all multiple points of the cover are the points of one line not belonging to the cover, since it is minimal. Translated to the dual situation we obtain that there exists a point $p \in Q(4, q)$ on which there are $q + 1$ lines each containing exactly two points of the blocking set. Proving the non-existence of such a blocking set we can proceed with the following steps.

step 1 Choose the special point of $Q(4, q)$ on which the multiple lines pass. Since the group of the quadric acts transitively on the points, we can choose this point $p_1$ arbitrary.

step 2 Now choose some points 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 group of the quadric to reduce the number of possibilities. Stabilizing the set $\{p_1, \ldots, p_k\}$, we have $n$ choices for $p_{k+1}$ if $n$ is the number of orbits of $\text{Stab}_{\{p_1, \ldots, p_k\}}(PGO(5, q))$ on the set of points of $Q(4, q)$ not collinear with $p_1, \ldots, p_k$. 

3
step 3 Choosing \( p_2, \ldots, p_k \) decreases the number of possible points for the blocking set on the \( q + 1 \) lines through \( p_1 \), since every \( p_i \) is collinear with exactly one point of every line on \( p_1 \). On each line we need exactly two points. Hence if there are \( n_i \) points on \( L_i \) (\( p_1 \in L_i, n_i \geq 2 \)) then we have \( \frac{n_i(n_i - 1)}{2} \) possibilities for each line \( L_i \), hence in total \( \prod_{i=1}^{q+1} \frac{n_i(n_i - 1)}{2} \) possibilities to extend the partial blocking set of step 2. If \( n_i < 2 \), then the configuration \( \{p_2, \ldots, p_k\} \) can be deleted. Hence we find the possible partial blocking sets \( \{p_2, \ldots, p_k, r_1, \ldots, r_{2(q+1)}\} \) which might be extendable with points not collinear with \( \{p_2, \ldots, p_k, r_1, \ldots, r_{2(q+1)}\} \).

step 4 Extend the partial blocking set but delete the configuration \( \{p_2, \ldots, p_k, r_1, \ldots, r_{2(q+1)}\} \) if \( k - 1 + 2(q + 1) + N < q^2 + 2 \), with \( N \) the number of remaining candidates. Do this extension using ordinary backtrack, if you add a point \( s_i \), then check the condition \( k - 1 + 2(q + 1) + i + N \geq q^2 + 2 \) again and stop the branch if this condition is not satisfied anymore.

Using this approach, we found that for both \( q = 5, 7 \), a minimal blocking set of \( \text{Q}(4, q) \) of size \( q^2 + 2 \) does not exist. Using the same ideas, we could also exclude the existence of a minimal blocking set of size \( q^2 + 3 \) if \( q = 7 \), 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 choose again a partial blocking set \( \{p_2, \ldots, p_k\} \), but we need now 3 points on every line 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 \( \{p_2, \ldots, p_k, r_1, \ldots, r_{3(q+1)}\} \). At last, we have to check the condition \( k - 1 + 3(q + 1) + N \geq q^2 + 3 \) while trying to extend the blocking set. Using our program we could indeed exclude the existence of such a minimal blocking set of \( \text{Q}(4, 7) \).

3 A computer search

The environment

We use GAP (Groups Algorithms and Programming) (see [8] for the newest version and technical information) on a Mosix cluster of 15 Linux computers. There are basically two reasons that explain our choice for GAP. At first, it provides a wide range of functions dealing with computations in finite fields and group theory. Together with the build in functions for vectors and list, they form an 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” is developed in two master thesis projects. ([14]). Together with the above-mentioned functions, it is possible to setup the problem with few commands, e.g., one can compute all points of an arbitrary quadric in coordinate representation in three commands. 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 [8]. For more information about ”pg”, its possibilities and the manual, we refer to [14].

We will now discuss the used GAP code.

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

```gap
RequirePackage("pg");
q := 5;
```
De Beule, Storme, Hoogewijs

P := ProjectiveGeometry(4,q);
Q := StandardParabolicQuadric(P);
G := StabilizerGroup(Q);
H := AsPermutationGroup(G);

These few lines set up the quadric completely. We will also store explicitly the points of the quadric in
two representations: as coordinates and as a natural number. More details can be found in the manual of
the package pg.

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

Since the stabilizer group of the quadric acts transitively on its generators (see [9]), 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
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 a natural number.

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 on a given point and the points collinear with a point. To omit
recalculations 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);
Add(collwithp,dummy);
od;
p_1 will be the special point on which the multiple lines pass. With the following code we also store in \( H \) the Stabilizer group of the point \( p_1 \). The point \( p_2 \) will be the first point of the blocking set. We can choose \( p_2 \) again since the stabilizer group of \( p_1 \) acts still transitively on the points not collinear with \( p_1 \).

\[
p_1 := \text{pointsn}[1];
\]
\[
\text{pointsn} := \text{Difference}(\text{pointsn}, \text{collwithp}[p_1]);
\]
\[
\text{cone} := \text{Difference}(\text{collwithp}[p_1],[p_1]);
\]
\[
\text{H} := \text{Stabilizer}(\text{H}, p_1);
\]
\[
\text{biglist} := [];
\]
\[
\text{keuzes} := [];
\]
\[
p_2 := \text{pointsn}[1];
\]

At this point, all initializations are done.

**Some basic functions**

The function `setup` computes some possible start configurations for the blocking set, as explained in step 2. We can adapt the parameters `param` and `sizecone`. The first parameter is the number of points our starting configuration will have. The second one is the minimum number of possible candidates to extend the partial blocking set with we need on the \( q + 1 \) lines on the point \( p_1 \).

\[
\text{setup} := \text{function}(\text{pointsn}, \text{keuze}, \text{H}, \text{keuzes}, \text{cone}, \text{param}, \text{sizecone}, \text{biglist})
\]
\[
\text{local npointsn, nH, nkeuzes, orbits, orbit, ncone;}
\]
\[
\text{npointsn} := \text{Difference}(\text{pointsn}, \text{collwithp}[\text{keuze}]);
\]
\[
\text{nkeuzes} := \text{Union}(\text{keuzes}, [\text{keuze}]);
\]
\[
\text{nH} := \text{Stabilizer}(\text{H}, \text{nkeuzes}, \text{OnSets});
\]
\[
\text{orbits} := \text{Orbits}(\text{nH}, \text{npointsn});
\]
\[
\text{ncone} := \text{Difference}(\text{cone}, \text{collwithp}[\text{keuze}]);
\]
\[
\text{if Length}(\text{nkeuzes}) = \text{param} \text{ then}
\]
\[
\text{if Length}(\text{ncone}) \geq \text{sizecone} \text{ then}
\]
\[
\text{Add}(\text{biglist}, \text{rec}(bs := \text{Set}(\text{nkeuzes}), \text{cone} := \text{ncone}, \text{length} := \text{Length}(\text{ncone})));
\]
\[
\text{fi;}
\]
\[
\text{else}
\]
\[
\text{for orbit in orbits do}
\]
\[
\text{setup}(\text{npointsn}, \text{orbit}[1], \text{H}, \text{nkeuzes}, \text{ncone}, \text{param}, \text{sizecone}, \text{biglist});
\]
\[
\text{od;}
\]
\[
\text{fi;}
\]
\[
\end;
\]

The function `analyselines` checks if, given all the points collinear with \( p_1 \), except the points which are collinear with the points from a given partial blocking set (the parameter `cone`), there are sufficient points (the parameter `length`) on every line 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`.

\[
\text{analyselines} := \text{function}(\text{cone}, \text{point}, \text{length})
\]
\[
\text{local bool, lines, i, l;}
\]
\[
\text{od;
\]
\[
\text{dummy := Set}(\text{dummy});
\]
\[
\text{collwithp}[i] := \text{dummy};
\]
\[
\text{od;
\]
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 start configurations with the function setup, we have to analyze them. Given de list with start configurations computed with setup, check that:

1 Are there sufficient points on the lines on the point $p_1$? If so:

2 Can we choose on every line at least the number of wanted points?

The variable biglist is a list with records containing all information. Such a record has a component $x$ containing the partial blocking set, cone contains the points collinear with $p_1$ which are not excluded by the choice of the points in $x$, point is the point $p_1$, sizecone is the total number of points of the blocking set on 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(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 start configuration described in step 2. The function is self-explanatory.

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
The second part of the code

After this code is executed, we can go further with the analyzed start configurations and compute a set of new start configurations as described in step 3. We will now discuss the code to execute this step. The first function is only of use in the big function `createstartconfs`.

```plaintext
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));
```
The third part of the code
The last function we need is a function which extends a partial blocking set as described in step 3 to a desired
blocking set (if possible), using an ordinary backtrack. The function is self-explanatory. If a blocking set
of the desired size is found, it is stored in a global variable resultlist.

```gap
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, sizebs, 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 first
part of the code is placed in a file functions2. After starting GAP, this file is read. Then we can execute
step 1 with two commands. We added the time command to give an idea of the computing time. This
command gives the computation time in milliseconds.

The case \( q = 5 \).
```gap
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);
```
At this point we executed step 1 and 2 for the case \( q = 5 \). It is clear that we go to step 3 with 21 start configurations. Going into step 3, we execute the following commands. We write the second part of the code in a file `functions3`:

```gap
gap> Read("./q45new/functions3");
gap> resultlist := [];
[ ]
gap> for param in [1..Length(result)] do
> x := result[param];
> 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;
> n := countbranches([x],p1,length);
> Print("number: ",param," 
");
> Print("length of resultlist: ",length(resultlist)," 
");
> Print("checked ",n," configurations\n");
> od;
```

We present the results in the following table:

<table>
<thead>
<tr>
<th>number</th>
<th>length of resultlist</th>
<th>checked configurations</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>162</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>81</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>36</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>162</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>9</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>81</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>9</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>9</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>36</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>81</td>
</tr>
<tr>
<td>11</td>
<td>0</td>
<td>729</td>
</tr>
<tr>
<td>12</td>
<td>0</td>
<td>81</td>
</tr>
<tr>
<td>13</td>
<td>0</td>
<td>9</td>
</tr>
<tr>
<td>14</td>
<td>0</td>
<td>81</td>
</tr>
<tr>
<td>15</td>
<td>0</td>
<td>81</td>
</tr>
<tr>
<td>16</td>
<td>0</td>
<td>9</td>
</tr>
<tr>
<td>17</td>
<td>0</td>
<td>9</td>
</tr>
<tr>
<td>18</td>
<td>0</td>
<td>81</td>
</tr>
<tr>
<td>19</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>20</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>21</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

```
gap> time;
gap> 810
Since the length of \texttt{resultlist} is 0, we have not 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 $q = 7$.

When $q = 7$, we will handle two cases: multiple lines that are blocked by two points and 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. In the first case, lines are blocked by two points so we need at least 16 points collinear with $p_1$ which are not excluded by the choice of the first 4 points. In the second case, lines are blocked by three points so we need at least 24 points. You can find these parameters back in the following code.

```gap
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 start configurations from step 2. Each one will give a new set start configurations after step 3. Some of these can be thrown away (like in the $q = 5$ case), for others, we may need 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 start configuration on which we execute step 2. We write the output in different files. If a start configuration must be held after step 3, we save the new workspace. In the file \texttt{functions3} we write the second part of the code. In the file \texttt{start} we write the following GAP commands:

```gap
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);

\[ \text{t := time;} \]
\[ \text{str := ".\!/q47new/results/workspaces/Wq47.create.";} \]
\[ \text{out := OutputTextString(str,true);} \]
\[ \text{PrintTo(out,String(param));} \]
\[ \text{CloseStream(out);} \]
\[ \text{n := countbranches([result[param]],p1,length);} \]
\[ \text{Print("number: ",param," 
");} \]
\[ \text{Print("length of resultlist: ",Length(resultlist)," 
");} \]
\[ \text{Print("checked ",n," configurations\n");} \]
\[ \text{Print("calculating time for confs: ",t," \n");} \]
\[ \text{if (Length(resultlist) = 0) then} \]
\[ \text{Print("quitting without saving workspace\n");} \]
\[ \text{quit;} \]
\[ \text{fi;} \]
\[ \text{Print("Saving Workspace\n");} \]
\[ \text{SaveWorkspace(str);} \]

A GAP job can be started with the following command.

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

We will give two examples of output files.

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> 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)
```
De Beule, Storme, Hoogewijs

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, ctblib, pg loaded.

In total 143 workspaces corresponding with start configurations from step 3 were saved. We can handle step 4 similar. The following GAP code, together with the function createbs, is put in a file start2

```gap
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.

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

We give the variable param the number of the configuration we want to handle. No blocking set of size 51 is found in all the configurations. 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> 1353
gap> function( pbs, pointsn, sizebs, choices, resultlist ) ... end
gap> gap> [ ]
gap> "./q47new/results/workspaces/results/Wq47.createbs."
gap> OutputTextString(50)
gap> gap> Length of resultlist: 1
gap> > > gap> 12270
gap> Length of biglist: 0
gap> Calculating time: 12270
gap> > Quitting without saving workspace

Executing step 3 took approximately 827697 seconds, executing step 4 took 1764 seconds.

At last 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 three we have to change two lines in the file start, the line initializing the size of the blocking set and the number of points 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 possibilities. 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, 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

To execute step 3 for all 3508 possibilities took 25272 seconds. As no configuration admits a blocking set, step 4 has not to be executed as in the $q = 5$ case. We conclude that a minimal blocking set of $Q(4, 7)$ of size 51 does not exist and that a minimal blocking set of $Q(4, 7)$ of size 52 with multiple lines blocked by three points does not exist.

**Acknowledgment**

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

**References**


