Advances in Mathematics of Communications
Volume 15, No. 2, 2021, 267–277
doi:10.3934/amc.2020065

# ON THE EXISTENCE OF PD-SETS: ALGORITHMS ARISING FROM AUTOMORPHISM GROUPS OF CODES

Nicola Pace*

Chair of Operations Research - Technische Universität München
Arcisstr. 21
80333 Munich, Germany

Angelo Sonnino

Dipartimento di Matematica, Informatica ed Economia - Università degli Studi della Basilicata
Viale dell'Ateneo Lucano, 10
85100 Potenza, Italy

(Communicated by Alfred Wassermann)

Abstract. This paper deals with the problem of determining whether a PD-set exists for a given linear code $\mathcal{C}$ and information set $I$. A computational approach is proposed and illustrated with two exceptional codes with automorphism groups isomorphic to the sporadic simple groups $\mathrm{M}_{12}$ and $\mathrm{M}_{22}$, respectively. In both cases, the existence of a PD–set is proven. In general, the algorithm works well whenever the code $\mathcal{C}$ has a very large automorphism group.

## 1. Introduction

In [45], F. J. MacWilliams developed a decoding method, called permutation decoding, that is feasible when the automorphism group contains a set of automorphisms, called a PD-set, with specific properties. A PD–set for a $t$–error-correcting code $C$ is a set $\mathcal{S}$ of automorphisms of $C$ such that every $t$–set of coordinate positions is moved by at least one member of $\mathcal{S}$ outside the information positions. Two fundamental problems arise:

(1) Determine whether $\mathrm{Aut}(C)$ contains a PD–set.
(2) If $\mathrm{Aut}(C)$ contains PD–sets, then find a PD-set of smallest possible size.

These problems have been studied by various authors for very particular classes of codes, see [7, 15, 16, 17, 19, 25, 27, 29, 30, 31, 32, 33, 34, 35, 42, 43, 44, 53, 57]. If the existence of a PD–set is known and the code has relatively small parameters, a computational approach to Problem (2) was proposed by the authors in [51]. This approach produced several examples of small PD–sets that, in some cases, can be proven to be the smallest possible. In general, it is convenient to start with a linear code with a large prescribed automorphism group and, not only because of permutation decoding, these codes have been widely investigated in the last years, see [2, 4, 6, 9, 8, 10, 11, 12, 13, 14, 18, 21, 24, 36, 37, 38, 39, 40, 41, 49, 50, 51, 52, 54, 56] and references therein.

2020 *Mathematics Subject Classification:* Primary: 94B05, 94B35; Secondary: 20B25.
*Key words and phrases:* Automorphism group, linear code, Mathieu group, permutation decoding, transitive code.

* Corresponding author: Nicola Pace.

This paper deals with Problem (1). Indeed, given a linear error-correcting code $C$ and an information set $I$, determining whether a PD–set exists is often computationally unfeasible, even for moderately large examples. A computational approach is proposed and illustrated with two exceptional codes. In the first case, a [77,10,32] binary code is considered. This code was constructed by A. Cossidente and the second author in [13] and has an exceptionally large automorphism group isomorphic to the Matheiu group $\mathrm{M}_{22}$. The second case is a ternary [66,10,36] code which was constructed by the first author in [49]. This code has the largest minimum distance among the ternary linear codes with length $n = 66$ and dimension $k = 10$; see [20]. Moreover, it has a large automorphism group isomorphic to $\mathrm{C}_2 \times \mathrm{M}_{12}$ and, in several ways, resembles the famous Golay code. In general, our algorithm works well whenever the code $C$ admits a large automorphism group.

Section 2 contains some preliminary notions on linear codes and permutation decoding. In Section 3 some group theoretical implications are presented. In Sections 4 and 5, the algorithm is presented and illustrated with the two examples.

## 2. Preliminaries

For a prime power integer $q$, let $V(n,q)$ be the $n$-dimensional vector space over the Galois field $\mathbb{F}_q$. Then, a $k$-dimensional subspace $C$ of $V(n,q)$ is a $q$-ary linear $[n,k]$-code. The integer $n$ is said to be the length of the code, while a vector $\mathbf{x} \in C$ is called a codeword; see [22, 46, 47].

A generator matrix for $C$ is a $k \times n$ matrix $M$ whose rows form a basis for $C$ as a subspace of $V(n,q)$. The Hamming distance $\mathrm{d}(\mathbf{x},\mathbf{y})$ between two vectors $\mathbf{x} = (x_1,\ldots,x_n)$ and $\mathbf{y} = (y_1,\ldots,y_n)$ is the number of $i$'s such that $x_i \neq y_i$, and the minimum distance of $C$ is the integer

$$d(C) = \min\{\, \mathrm{d}(\mathbf{x},\mathbf{y}) \mid \mathbf{x},\mathbf{y} \in C;\ \mathbf{x} \neq \mathbf{y} \,\}.$$

The Hamming weight $w(\mathbf{x})$ of a vector $\mathbf{x}$ is the number of its nonzero components, and the minimum weight $w(C)$ of $C$ is the minimum weight of its codewords different from the zero vector. For any linear code $C$ the minimum distance $d(C)$ and minimum weight $w(C)$ coincide.

The minimum distance $d(C)$ of a code $C$ is related to its ability to correct errors. Indeed, using the maximum likelihood decoding method, a code $C$ can correct up to $t = \lfloor \frac{d(C)-1}{2} \rfloor$ errors. In this case, we will say that $C$ is a $t$-error correcting code.

Let $C_1$ and $C_2$ be two $q$-ary $[n,k]$-linear codes with generator matrices $M_1$ and $M_2$ respectively. Then $C_1$ and $C_2$ are said to be equivalent if and only if

$$M_1 B = M_2$$

for some $n \times n$ monomial matrix $B$. The group of linear transformations on ${F_q}^n$ which preserves the Hamming weight is the monomial group

$$\mathcal{M}_n(F_q) = \left(\mathbb{F}_q \setminus \{0\}\right)^n \rtimes \mathrm{Sym}(n), \tag{1}$$

where the product in $\left(\mathbb{F}_q \setminus \{0\}\right)^n$ is $(a_1,\ldots,a_n)(b_1,\ldots,b_n) = (a_1b_1,\ldots,a_nb_n)$. Given $\mathbf{a} = (a_1,\ldots a_n) \in \left(\mathbb{F}_q \setminus \{0\}\right)^n$ and $\sigma \in \mathrm{Sym}(n)$, for any codeword $\mathbf{x} = (x_1,\ldots,x_n)$ write

$$(\mathbf{a},\sigma)(\mathbf{x}) = (a_{\sigma^{-1}(1)}x_{\sigma^{-1}(1)},\ldots,a_{\sigma^{-1}(n)}x_{\sigma^{-1}(n)}).$$

The linear automorphism group $\mathrm{Aut}(C)$ of a $q$-ary $[n,k]$-linear code $C$ is the group of linear transformations in $\mathcal{M}_n(F_q)$ which leaves the code $C$ invariant.

2.1. PD-sets. For a $t$-error correcting $[n,k,d]$ linear code $C$ over $\mathbb{F}_q$, whose generator matrix we suppose to be in standard form $G = (\mathrm{I}_k|A)$, let $I$ be the set of information positions and $J$ the set of check positions (the redundancy). A set $\mathcal{S}$ consisting of elements of the group $\mathrm{Aut}(C)$ is called a permutation decoding set (for short a PD-set) for $C$ if for every subset $B$ of $\{1,\dots,n\}$, with $|B| = t \leq \left\lfloor \frac{d-1}{2} \right\rfloor$, there exists an element $(\mathbf{a},\sigma) \in \mathcal{S}$ such that $\sigma(B) \cap I = \emptyset$; see [23, Page 1413]. In other words, a PD-set for an error correcting code $C$ with parameters $[n,k,d]$ is a set $\mathcal{S}$ consisting of automorphisms of the code such that every possible error vector of weight at most $t$ can be moved outside the information part by some element of $\mathcal{S}$.

The permutation decoding algorithm is fairly simple once a PD-set is found. For instance, suppose that $C$ is a $t$-error correcting $[n,k,d]$ linear code over $\mathbb{F}_q$ with both the generator matrix $G = (\mathrm{I}_k|A)$ and the parity-check matrix $H = (-{}^TA|I_{n-k})$ in standard form, so that we have $I = \{1,\dots,k\}$ and $J = \{k+1,\dots,n\}$. Further, let $\mathcal{S} = \{\gamma_1,\dots,\gamma_m\}$ be a PD-set for $C$. In this setting, any message defined by a $k$-ple $\mathbf{m}$ is encoded as $\mathbf{c} = \mathbf{m}G = (c_1,\dots,c_n)$ with the information symbols in the first $k$ positions.

Suppose now that a codeword $\mathbf{c}$ is sent, but $\mathbf{r} = \mathbf{c} + \mathbf{e}$ is received, with the error vector $\mathbf{e}$ having weight at most $t$. Compute the syndromes ${}^T\mathbf{s}_i = H \cdot {}^T\gamma_i(\mathbf{r})$ for $1 \leq i \leq m$ until an index $i$ is found such that the weight of $\mathbf{s}_i$ is $t$ or less. Use the information symbols that are in $\gamma_i(\mathbf{r})$ to identify a codeword $\mathbf{c}'$ carrying the same information symbols in the first $k$ positions. Then $\mathbf{r}$ can be decoded as $\mathbf{c} = \gamma_i^{-1}(\mathbf{c}')$.

That this algorithm actually works is granted by the following result; see [23, Theorem 8.1] and also [25, Result 2].

**Result 1.** *Let $C$ be a $t$-error correcting $[n,k,d]$ linear code with parity check matrix $H = (-{}^TA|I_{n-k})$ in standard form. If $\mathbf{r} = \mathbf{c} + \mathbf{e}$ is a vector with $\mathbf{c} \in C$ and $\mathrm{w}(\mathbf{e}) \leq t$, then the information symbols in $\mathbf{r}$ are correct if and only if the weight of the syndrome ${}^T\mathbf{s} = H \cdot {}^T\mathbf{r}$ of $\mathbf{r}$ does not exceed $t$.*

We remark that the whole permutation decoding procedure works since, for $\gamma \in \mathrm{Aut}(C)$, if $\mathbf{r} = \mathbf{c} + \mathbf{e}$ then $\gamma(\mathbf{r}) = \gamma(\mathbf{c}) + \gamma(\mathbf{e})$.

In [27], the notion of partial permutation decoding was introduced, where an $s$-PD-set is a set of automorphisms that can correct up to $s$ errors, where $s \leq t$. More precisely, an $s$-PD-set is a set $\mathcal{S}$ of automorphisms of $C$ which is such that every $s$-set of coordinate positions is moved by at least one member of $\mathcal{S}$ into the check positions $J$.

In general, the problem of constructing PD-sets for error correcting codes is a very hard one. Further, the existence of a PD-set may also depend on the way the information set is chosen, therefore the existence of PD-sets is not invariant under equivalence of codes. More than that, for many known codes a PD-set may not even exist. On the other hand, the larger the automorphism group of a code, the more likely it is for a PD-set to exist; see [23, 42, 43, 44]. Codes with large prescribed automorphism groups have been constructed in [51] by selecting suitable matrix group representations from the Atlas of Finite Groups Representations [1] and considering the corresponding transitive codes.

It should also be noted that the decoding algorithm is more efficient when the PD-set is as small as possible. A lower bound for the size of a PD-set is established by the following result; see [19, 23].

**Result 2** (Gordon). *Let $C$ be a $t$-error correcting $[n,k]$ linear code with redundancy $r=n-k$, and $\mathcal{S}$ a PD-set for $C$. Then*

$$|\mathcal{S}| \geq \left\lceil \frac{n}{r} \left\lceil \frac{n-1}{r-1} \cdots \left\lceil \frac{n-t+1}{r-t+1} \right\rceil \cdots \right\rceil \right\rceil .$$

Under certain conditions, PD-sets can be conveniently described in group-theoretical terms. The next section will be devoted to this approach.

## 3. Group-theoretical implications

Let $G$ be a permutation group acting on a set $X$ and, for an integer $t \leq |X|$, set $\binom{X}{t} = \{\, T \subset X \mid |T| = t \,\}$.

**Proposition 1.** *Let $C$ be a $t$-error correcting $[n,k]$ linear code with a set of information positions denoted by $I$ and $G$ an automorphism group of $C$. Then, the following statements are equivalent.*

(a) *$C$ admits a $t$–PD set.*
(b) *For every $T \in \binom{X}{t}$, there exists $g \in G$ such that $T \cap I^g = \emptyset$.*
(c) *For every $T \in \binom{X}{t}$, there exist $g, h \in G$ such that $T^h \cap I^g = \emptyset$.*

*Proof.* If (a) holds then there is an element $h \in G$ such that $T^h \cap I = \emptyset$, that is, $(T^h \cap I)^{h^{-1}} = T \cap I^{h^{-1}} = \emptyset$. Hence (a) implies (b) with $g = h^{-1}$. To prove that (b) implies (c) it suffices to take $h = \mathrm{id}_G$. Finally, if (c) holds then for every $T \in \binom{X}{t}$, there exist $g, h \in G$ such that $T^h \cap I^g = \emptyset$. Hence, $hg^{-1}$ moves every element of $T$ outside the set of information positions $I$. This completes the proof. □

We recall a classical theorem from group theory [5, 48] which will be useful for the construction of PD-sets. Let $G$ be a permutation group acting on a nonempty set $\Omega$. Let

$$x^G = \{\, x^g \mid g \in G \,\}$$

denote the orbit of an element $x \in \Omega$ under the action of the group $G$ and for any $g \in G$ set

$$\mathrm{Fix}(g) = \{\, x \in \Omega \mid x^g = x \,\}.$$

**Result 3** (Burnside's lemma). *If $t$ is the number of orbits under the action of $G$ on $\Omega$, then*

$$t\,|G| = \sum_{g \in G} |\mathrm{Fix}(g)|.$$

## 4. Algorithms

Henceforth $X = \{1, \ldots, n\}$ will denote the set of positions in codewords of a code $C$, so that the information set $I$ is just a $k$-subset of $X$.

### 4.1. Naïve Algorithms.

Fix $G$, $I$ and $t$. A key problem is that of determining whether $G$ admits a $t$–PD set or not. Besides the most obvious sufficient conditions, such as $G$ being $t$-transitive, very little is known about this decisional problem. For instance, the sufficient condition $kt < n$ holds for cyclic codes [45] and a slightly

more general result can be found in [28, Lemma 7]. The simplest naïve algorithm is the following.

**Data:** $G$, $I$, $t$
**Result:** "*Success*","*Fail*"
**for** $T \in \binom{X}{t}$ **do**
  **if** $T^g \cap I \neq \emptyset$, *for all* $g \in G$ **then**
    **return** *"Fail";*
  **end**
**end**
**return** *"Success";*

**Algorithm 1:** Naïve Algorithm

A first improvement to the naïve algorithm can be done by precomputing the set $I^G = \{ I^g \mid g \in G \}$. Then, using part (b) of Proposition 1, one could twist the problem and simply check that, for every $T \in \binom{X}{t}$, there exists a $U \in I^G$ such that $T \cap U = \emptyset$. The clear advantage is that it is not necessary to compute $T^g$ for all $g \in G$ but only $I^G$, and this is done only once.

**Data:** $G$, $I$, $t$
**Result:** "*Success*","*Fail*"
Compute the set $I^G = \{ I^g \mid g \in G \}$;
**for** $T \in \binom{X}{t}$ **do**
  **if** $T \cap U \neq \emptyset$, *for all* $U \in I^G$ **then**
    **return** *"Fail";*
  **end**
**end**
**return** *"Success";*

**Algorithm 2:** Precomputing the set $I^G$.

Unfortunately, even for relatively small examples, the set $\binom{X}{t}$ is often too large and this makes the computation unfeasible. By using part (c) of Proposition 1, one can restrict consideration to $G$-orbit representatives of $t$-subsets rather than the whole $\binom{X}{t}$. This leads to the following algorithm.

**Data:** $G$, $I$, $t$
**Result:** "*Success*","*Fail*"
Compute the set $I^G = \{ I^g \mid g \in G \}$;
Consider $G$-orbits of $t$–subsets of $X$ and determine orbits representatives:
$T_1, \ldots, T_\ell$;
**for** $i \in \{1, \ldots, \ell\}$ **do**
  **if** $U \cap T_i \neq \emptyset$, *for all* $U \in I^G$ **then**
    **return** *"Fail";*
  **end**
**end**
**return** *"Success";*

**Algorithm 3:** Precomputing the set $I^G$ and use orbits of $t$–subsets of $X$.

If it is possible to compute and store all orbit representatives $T_1, \ldots, T_\ell$, this is possibly a good way to determine whether $G$ admits a $t$–PD set or not. If the code

is short, or $t$ is small, this can be easily done. However, in general this computation is known to be out of reach, even for moderately large values of $n$ and $t$.

4.2. A MORE EFFICIENT ALGORITHM. In this section, a technique is described which considerably shortens the computation time when it is not possible to compute the whole set of orbit representatives $T_1 \dots T_\ell$ as in Algorithm 3.

Indeed, a suitable value $t_0 < t$ can be chosen in such a way that it is feasible to compute all $G$–orbit representatives of $t_0$–subsets of $X$, which can be estimated by Theorem 3. Every $G$-orbit of $t$-subsets contains a representative of type $T_{0,i} \cup V$, for $i \in \{1, \dots, \ell\}$ and $V \in \binom{X \setminus T_{0,i}}{t-t_0}$. Therefore, the algorithm works since it actually tests all $G$-orbits of $t$-subsets. Note that although some orbits may be tested more than once, this algorithm seems to be the best solution when the computational complexity using Algorithm 3 turns out to be too high.

**Data:** $G$, $I$, $t$, $t_0$
**Result:** "*Success*","*Fail*"
Compute the set $I^G = \{ I^g \mid g \in G \}$;
Consider $G$-orbits of $t_0$–subsets of $X$ and determine orbits representatives:
$T_{0,1}, \dots, T_{0,\ell}$;
**for** $i \in \{1, \dots, \ell\}$ **do**
  Compute $I_{0,i} = \{ U \mid U \in I^g,\ U \cap T_{0,i} = \emptyset \}$;
  **for** $V \in \binom{X \setminus T_{0,i}}{t-t_0}$ **do**
    **if** $U \cap V \neq \emptyset$, *for all* $U \in I_{0,i}$ **then**
      **return** *"Fail";*
    **end**
  **end**
**end**
**return** *"Success";*

**Algorithm 4:** Algorithm implemented

## 5. EXAMPLES

The Algorithm 4 was implemented in two separate parts, using the Computer Algebra System Magma [3] and the programming language C++ [55]. More precisely, the sets $I^G$ and $T_{0,1}, \dots, T_{0,\ell}$ were precomputed using Magma and the core of the algorithm was implemented using the programming language C++. The core of the algorithm ran in parallel on two machines:

- Apple iMac with a 3.4 GHz Intel Core i7 processor.
- Dell Precision T7610 with a Dual Intel Xeon Processor E5-2680 v2 (Ten Core HT, 2.8 GHz Turbo, 25 MB).

In particular, two technical details were crucial for the feasibility of the computation.

- For each $i$, the set $I_{0,i} = \{ U \mid U \in I^g, U \cap T_{0,i} = \emptyset \}$ is computed, and those sets that are certainly not going to work are removed from $I^G$. This is crucial because otherwise it would be necessary to test disjointness for all the sets $V \in \binom{X \setminus T_{0,i}}{t-t_0}$ with all elements in $I^G$.
- The internal cycle "**for**" tests whether $V$ is disjoint with every set $U \in I_{0,i}$. Since there are faster algorithms for sorted data, it is convenient to deal with sorted sets $U$ and $V$. In practice, this is not an obstacle because $I^g$ is computed

only once and, in the iteration "**for** $V \in \binom{X \setminus T_{0,i}}{t-t_0}$", the set $V$ can easily be constructed as already sorted.

5.1. A linear code with automorphism group $\mathrm{M}_{22}$. In [13], A. Cossidente and the second author constructed a [77,10,32] binary code admitting the automorphism group $\mathrm{M}_{22}$. Since the existence of a PD–set may depend on the choice of the information set, we have to provide an explicit description of the code, its information set $I$ and automorphism group $G$. A generator matrix $M$ for this code with information set $I = \{1, \ldots, 10\}$ is the following:

$$M = \begin{pmatrix}
10000000001101111010111001101101111111001111001001001101001010110011000010100\\
01000000001111010111011100000100101010011100010010011100011000000010101101100\\
00100000001000000111100110110100111110101011000010010010000010011010100111010\\
00010000001111011011100000011010001110100001111000011100010010010000011001\\
00001000000100110000110110000101101011000111010001010010011100010101101011010\\
00000100000010110000110000011000111110110011001110101110011110010011111111111\\
00000010000001110110110000000011000011111001101110011001001000101011001100101\\
00000001000000001110000001111011001000111101011100111110001101101101111111111\\
00000000100000000011100011111111111011011100010111000101101010100100010000000\\
00000000010000000000001111111000111111001000111010001101101110111111110111111
\end{pmatrix}.$$

One can consider the automorphism group $G = \langle a_1, a_2 \rangle \cong \mathrm{M}_{22}$, where $|G| =$ 443,520 and

$$\begin{aligned}
a_1 = {} & (1,15)(2,72,74,27,9,32,10,45)(3,63,50,68,56,66,77,16)\\
& (4,75,11,76,26,60,48,8)(5,17,46,65,53,43,7,38)(6,29,59,25)\\
& (12,44,13,28,14,34,70,31)(18,37,21,71,67,36,42,41)\\
& (19,39,22,33)(20,64,51,62,40,24,69,30)\\
& (23,52,73,58,35,61,57,47)(54,55);\\
a_2 = {} & (6,13)(8,17)(9,35)(10,23)(15,16)(18,19)(20,60)(21,47)(22,41)\\
& (24,25)(26,43)(27,50)(28,58)(29,30)(31,37)(32,49)(33,51)\\
& (34,42)(36,61)(38,39)(40,48)(44,52)(45,56)(46,66)(53,63)\\
& (55,57)(62,64)(67,71).
\end{aligned}$$

The Algorithm 4 was implemented using the parameter $t_0 = 8$. It turns out that there are 26,859 $\mathrm{M}_{22}$–orbits of 8–subsets of $X = \{1, \ldots, 77\}$. The orbits $T_1, \ldots, T_{26,859}$ were computed using the software Magma running on a standard Personal Computer. The software implemented in C++ ran for approximately three days on the workstation Dell Precision T7610. The existence of a PD–set was proven.

5.2. A linear code with automorphism group $\mathrm{M}_{12}$. In [49], the first author constructed a [66,10,36] ternary code admitting the automorphism group $\mathrm{C}_2 \times \mathrm{M}_{12}$. A generator matrix $M$ for this code having information set $I = \{1, \ldots, 10\}$ is the following:

$$M = \begin{pmatrix}
10000000002100201200102011210022002221100221201121102210100022020 2\\
01000000001211112002122212002000212222201211101121110222122002212 2\\
00100000001102102000110012200212110111211202200220020222102120010 0\\
00010000002110110020021021021012211002010112012110200101211220100 0\\
00001000000211120212021220010010112001022010101100112000112221022 0\\
00000100001211222021121221122012101220112111102002222121002020222 0\\
00000010002002002000100221020102212020121100101212102020102221212 1\\
00000001000121020110001222002010201111021221201012011010200102210\\
00000000100222001022010220012002022200011121101120102012110202110\\
00000000012010112020122221011220001221110000111101120010010101202 0
\end{pmatrix}.$$

One can consider the permutation part $G = \langle a_1, a_2 \rangle \cong \mathrm{M}_{12}$ of the automorphism group of the codes, where $|G| = 95{,}040$ and

$$\begin{aligned} a_1 = {} & (3,23,44,34,40,61,60,21)(4,12,32,26,55,45,17,27) \\ & (5,20,13,58,29,9,6,16)(7,14,66,64)(8,42,43,28,41,56,59,47) \\ & (10,24,48,54,57,50,18,33)(11,22)(15,39,31,52,53,65,62,36) \\ & (19,46)(25,51,49,37,35,38,63,30); \\ a_2 = {} & (1,7)(2,42)(4,5)(6,40)(8,65)(9,14)(10,45)(11,44)(12,55)(13,18) \\ & (15,32)(16,51)(17,22)(19,27)(20,28)(21,46)(24,30)(26,53)(31,35) \\ & (33,47)(34,49)(36,43)(37,59)(38,41)(39,50)(48,61)(52,58)(54,64) \\ & (56,66)(60,63). \end{aligned}$$

The Algorithm 4 was implemented using the parameter $t_0 = 7$. It turns out that there are 8810 $\mathrm{M}_{12}$–orbits of 7–subsets of $X = \{1, \ldots, 66\}$. The orbits $T_1, \ldots, T_{8810}$ were computed using the software Magma running on a standard Personal Computer. The software implemented in C++ ran in parallel on our two workstations for approximately 20 days. The existence of a PD–set was proven.

## 6. Conclusions

A practical problem in permutation decoding is to determine whether the automorphism group of a code contains a PD-set. In general, the solution of this decisional problem is out of reach, even for moderately large values of $n$ and $t$. To the best of our knowledge, the only algorithm known for a generic code was the naïve algorithm (Algorithm 1). In previous works, the existence of PD–sets (or, more often, partial PD-sets) was proven only in very particular cases with ad-hoc arguments and exploiting the particular structure of the codes; see, for instance, [7, 15, 19, 26, 27, 32, 42, 53, 57].

Algorithm 4 can be used to efficiently test codes admitting large automorphism groups, which are the best candidates for permutation decoding. The idea of the algorithm is based on Proposition 1, and a key step is determining a suitable value $t_0$. In particular, the value $t_0$ should be as "large as possible" while keeping the value $\ell$ to be "not too big" with respect to the available computational resources. In our implementation, the existence of a PD–set was proven for two exceptional linear codes [13, 49] in a reasonable time and using two standard workstations.

In Section 4.2 we underlined the fact that certain orbits are possibly tested more than once. Although we do not see how this can be avoided, there could be room for improvement. In this respect, exploiting structural properties that are peculiar to a particular group or linear code may produce a further optimisation of Algorithm 4 for certain classes of codes.

## Acknowledgments

N. Pace would like to thank the School of Mathematical Sciences, Dublin Institute of Technology, for granting access to the workstation Dell Precision T7610. A considerable amount of computation needed for this research was carried out on this workstation. A. Sonnino was partially supported by the Italian Ministero dell'Istruzione, dell'Università e della Ricerca (MIUR) within the PRIN Project No. 2012XZE22K_006.

## References


[1] R. Abbott, J. Bray, S. Linton, S. Nickerson, S. Norton, R. Parker, I. Suleiman, J. Tripp, P. Walsh and R. Wilson, Atlas of finite group representations - version 3, http://brauer.maths.qmul.ac.uk/Atlas/.

[2] J. Bierbrauer, S. Marcugini and F. Pambianco, The Pace code, the Mathieu group $M_{12}$ and the small Witt design $S(5,6,12)$, *Discrete Math.*, **340** (2017), 1187–1190.

[3] W. Bosma, J. J. Cannon and C. Playoust, The Magma algebra system. I. The user language, *J. Symbolic Comput.*, **24** (1997), 235–265.

[4] M. Braun, A. Kohnert and A. Wassermann, Optimal linear codes from matrix groups, *IEEE Trans. Inform. Theory*, **51** (2005), 4247–4251.

[5] W. Burnside, *Theory of Groups of Finite Order*, 2nd edition, Cambridge University Press, Cambridge, 1911.

[6] P. Camion, Linear codes with given automorphism groups, *Discrete Math.*, **3** (1972), 33–45.

[7] H. Chabanne, Permutation decoding of abelian codes, *IEEE Trans. Inform. Theory*, **38** (1992), 1826–1829.

[8] A. Cossidente, C. Nolè and A. Sonnino, Cap codes arising from duality, *Bull. Inst. Combin. Appl.*, **67** (2013), 33–42.

[9] A. Cossidente and A. Sonnino, A geometric construction of a $[110,5,90]_9$–linear code admitting the Mathieu group $M_{11}$, *IEEE Trans. Inform. Theory*, **54** (2008), 5251–5252.

[10] A. Cossidente and A. Sonnino, Finite geometry and the Gale transform, *Discrete Math.*, **310** (2010), 3206–3210.

[11] A. Cossidente and A. Sonnino, Some recent results in finite geometry and coding theory arising from the Gale transform, *Rend. Mat. Appl. (7)*, **30** (2010), 67–76.

[12] A. Cossidente and A. Sonnino, Linear codes arising from the Gale transform of distinguished subsets of some projective spaces, *Discrete Math.*, **312** (2012), 647–651.

[13] A. Cossidente and A. Sonnino, On graphs and codes associated to the sporadic simple groups HS and $M_{22}$, *Australas. J. Combin.*, **60** (2014), 208–216.

[14] D. Crnković, S. Rukavina and L. Simčić, Binary doubly-even self-dual codes of length 72 with large automorphism groups, *Math. Commun.*, **18** (2013), 297–308.

[15] W. Fish, Binary codes and partial permutation decoding sets from the Johnson graphs, *Graphs Combin.*, **31** (2015), 1381–1396.

[16] W. Fish, J. D. Key and E. Mwambene, Partial permutation decoding for simplex codes, *Adv. Math. Commun.*, **6** (2012), 505–516.

[17] W. Fish, K. Kumwenda and E. Mwambene, Codes related to line graphs of triangular graphs and permutation decoding, *Quaest. Math.*, **35** (2012), 489–505.

[18] M. Giulietti, G. Korchmáros, S. Marcugini and F. Pambianco, Transitive $A_6$-invariant $k$-arcs in $PG(2,q)$, *Des. Codes Cryptogr.*, **68** (2013), 73–79.

[19] D. M. Gordon, Minimal permutation sets for decoding the binary Golay codes, *IEEE Trans. Inform. Theory*, **28** (1982), 541–543.

[20] M. Grassl, *Bounds on the Minimum Distance of Linear Codes and Quantum Codes*, Online available at http://www.codetables.de. Accessed on December 22, 2020.

[21] R. Hill, On the largest size of cap in $s_{5,3}$, *Atti Accad. Naz. Lincei Rend. Cl. Sci. Fis. Mat. Natur.*, **54** (1974), 378–384.

[22] R. Hill, *A First Course in Coding Theory*, Oxford Applied Mathematics and Computing Science Series, The Clarendon Press, Oxford University Press, New York, 1986.

[23] W. C. Huffman, Codes and groups, in *Handbook of Coding Theory* (eds. V. S. Pless and W. C. Huffman), vol. 2, part 2, North-Holland, Amsterdam, 1998, chapter 17, 1345–1440.

[24] L. Indaco and G. Korchmáros, 42-arcs in $PG(2,q)$ left invariant by $PSL(2,7)$, *Des. Codes Cryptogr.*, **64** (2012), 33–46.

[25] J. D. Key, Permutation decoding for codes from designs, finite geometries and graphs, in *Information Security, Coding Theory and Related Combinatorics* (eds. D. Crnkovič and V. Tonchev), vol. 29 of NATO Sci. Peace Secur. Ser. D Inf. Commun. Secur., IOS, Amsterdam, 2011, 172–201.

[26] J. D. Key and J. Limbupasiriporn, Permutation decoding of codes from Paley graphs, *Congr. Numer.*, **170** (2004), 143–155.

[27] J. D. Key, T. P. McDonough and V. C. Mavron, Partial permutation decoding for codes from finite planes, *European J. Combin.*, **26** (2005), 665–682.

[28] J. D. Key, T. P. McDonough and V. C. Mavron, Information sets and partial permutation decoding for codes from finite geometries, *Finite Fields Appl.*, **12** (2006), 232–247.
[29] J. D. Key, J. Moori and B. G. Rodrigues, Permutation decoding for the binary codes from triangular graphs, *European J. Combin.*, **25** (2004), 113–123.
[30] J. D. Key, J. Moori and B. G. Rodrigues, Binary codes from graphs on triples and permutation decoding, *Ars Combin.*, **79** (2006), 11–19.
[31] J. D. Key, J. Moori and B. G. Rodrigues, Partial permutation decoding of some binary codes from graphs on triples, *Ars Combin.*, **91** (2009), 363–371.
[32] J. D. Key, J. Moori and B. G. Rodrigues, Codes associated with triangular graphs and permutation decoding, *Int. J. Inf. Coding Theory*, **1** (2010), 334–349.
[33] J. D. Key and B. G. Rodrigues, Codes from lattice and related graphs, and permutation decoding, *Discrete Appl. Math.*, **158** (2010), 1807–1815.
[34] J. D. Key and P. Seneviratne, Binary codes from rectangular lattice graphs and permutation decoding, *European J. Combin.*, **28** (2007), 121–126.
[35] J. D. Key and P. Seneviratne, Permutation decoding for binary codes from lattice graphs, *Discrete Math.*, **308** (2008), 2862–2867.
[36] W. Knapp and H.-J. Schaeffer, On the codes related to the Higman-Sims graph, *Electron. J. Combin.*, **22** (2015), Paper 1.19, 58 pp.
[37] W. Knapp and P. Schmid, Codes with prescribed permutation group, *J. Algebra*, **67** (1980), 415–435.
[38] A. Kohnert, Constructing two-weight codes with prescribed groups of automorphisms, *Discrete Appl. Math.*, **155** (2007), 1451–1457.
[39] A. Kohnert and A. Wassermann, Construction of binary and ternary self-orthogonal linear codes, *Discrete Appl. Math.*, **157** (2009), 2118–2123.
[40] A. Kohnert and J. Zwanzger, New linear codes with prescribed group of automorphisms found by heuristic search, *Adv. Math. Commun.*, **3** (2009), 157–166.
[41] G. Korchmáros and N. Pace, Infinite family of large complete arcs in $\mathrm{PG}(2, q^n)$, with $q$ odd and $n > 1$ odd, *Des. Codes Cryptogr.*, **55** (2010), 285–296.
[42] H.-J. Kroll and R. Vincenti, PD-sets for the codes related to some classical varieties, *Discrete Math.*, **301** (2005), 89–105.
[43] H.-J. Kroll and R. Vincenti, Antiblocking systems and PD-sets, *Discrete Math.*, **308** (2008), 401–407.
[44] H.-J. Kroll and R. Vincenti, PD-sets for binary RM-codes and the codes related to the Klein quadric and to the Schubert variety of $\mathrm{PG}(5, 2)$, *Discrete Math.*, **308** (2008), 408–414.
[45] F. J. MacWilliams, Permutation decoding of systematic codes, *Bell System Tech. J.*, **43** (1964), 485–505.
[46] F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error-Correcting Codes. I*, North-Holland Publishing Co., Amsterdam-New York-Oxford, 1977, North-Holland Mathematical Library, Vol. 16.
[47] F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error-Correcting Codes. II*, North-Holland Publishing Co., Amsterdam-New York-Oxford, 1977, North-Holland Mathematical Library, Vol. 16.
[48] P. M. Neumann, A lemma that is not Burnside's, *Math. Sci.*, **4** (1979), 133–141.
[49] N. Pace, New ternary linear codes from projectivity groups, *Discrete Math.*, **331** (2014), 22–26.
[50] N. Pace, On small complete arcs and transitive $A_5$-invariant arcs in the projective plane $PG(2, q)$, *J. Combin. Des.*, **22** (2014), 425–434.
[51] N. Pace and A. Sonnino, On linear codes admitting large automorphism groups, *Des. Codes Cryptogr.*, **83** (2017), 115–143.
[52] B. G. Rodrigues, Self-orthogonal designs and codes from the symplectic groups $s4(3)$ and $s4(4)$, *Discrete Math.*, **308** (2008), 1941–1950.
[53] P. Seneviratne, Codes associated with circulant graphs and permutation decoding, *Des. Codes Cryptogr.*, **70** (2014), 27–33.
[54] A. Sonnino, Transitive $\mathrm{PSL}(2, 7)$-invariant 42-arcs in 3-dimensional projective spaces, *Des. Codes Cryptogr.*, **72** (2014), 455–463.
[55] B. Stroustrup, *The C++ Programming Language*, 4th edition, Addison-Wesley, Upper Saddle River, NJ, 2013.

[56] L. M. G. M. Tolhuizen and W. J. van Gils, A large automorphism group decreases the number of computations in the construction of an optimal encoder/decoder pair for a linear block code, *IEEE Trans. Inform. Theory*, **34** (1988), 333–338.
[57] J. Wolfmann, A permutation decoding of the (24, 12, 8) Golay code, *IEEE Trans. Inform. Theory*, **29** (1983), 748–750.

Received February 2017; revised June 2018.

*E-mail address*: nicolaonline@libero.it
*E-mail address*: angelo.sonnino@unibas.it