Lecture Notes in Management Science (2016) Vol. 8, 86–92 ISSN 2008-0050 (Print), ISSN 1927-0097 (Online)

Packing regularity of permutation codes

János Barta [1], Roberto Montemanni [1] and Derek H. Smith [2]
[1] Dalle Molle Institute for Artificial Intelligence (IDSIA – USI/SUPSI), Galleria 2, 6928 Manno, Switzerland
{janos.barta, roberto.montemanni}@supsi.ch
[2] Division of Mathematics and Statistics, University of South Wales, Pontypridd, CF37 1DL, Wales, UK
derek.smith@southwales.ac.uk

Proc. ICAOR 2016
Rotterdam, The Netherlands

Keywords:
Coding theory
Combinatorial optimization
Discrete geometry
Permutation codes
Sphere packing

Abstract

Permutation codes have been extensively investigated, both because of their intrinsic mathematical interest and because of relevant applications based on error-correcting codes. The Maximum Permutation Code Problem (MPCP) is a challenging packing problem on permutations. The objective is to maximize the size of permutation codes with a given minimum Hamming distance between the codewords. In a similar way to the well-known sphere packing problem, an optimal permutation packing usually has a highly regular structure. In this paper a new idea of regularity degree of permutation codes is developed and the relationship between packing density and regularity degree of permutation codes is investigated. Computational experiments on random permutation packings run on different MPCPs confirm that, analogously to the sphere packing problem, the regularity degree of permutation codes tends to increase as the code size approaches to the optimum.

Introduction

For several centuries mathematicians have been fascinated by combinatorial problems related to permutations. Leonhard Euler, the great mathematician of the 18th century, was one of the first who systematically studied permutations and in particular latin squares. Two centuries later, when the works of Shannon and Hamming founded the subject of Information Theory, new challenging combinatorial problems emerged. In particular, a crucial role is played by error-correcting codes. These are codes capable of detecting and correcting errors during data transmission. The ability to identify and correct errors is strongly related to the minimum Hamming distance between the codewords, i.e. the minimum number of differing components in the codewords. Among the optimization problems concerning various error-correcting codes the Maximum Permutation Code Problem (MPCP) is enjoying particular attention, because of its potential applications to powerline communication (PLC) (see Colbourn *et al* (2004), Han Vinck (2000) and Pavlidou *et al* (2003)). Since one of the main concerns in PLC is interference due to noise, a reliable error-correcting coding protocol is required. One of these is the so-called M-ary frequency shift keying (FSK), in which a finite number of frequencies is used to modulate the signal. Permutation codes arise as a suitable mathematical framework for FSK. Furthermore, permutation codes have been applied also for coding with block ciphers and for the design of multilevel flash memories (see de la Torre *et al* (2000) and Jiang *et al* (2008)). From a mathematical perspective the MPCP represents a fascinating link between coding theory, group theory and combinatorics. The MPCP consists of finding the largest set of permutations of length n, such that the Hamming distance between the permutations is at least equal to a fixed value d. Several different methods have been developed to study and solve the MPCP, such as linear programming (Bogaerts (2010), Tarnanen (1999)), group theory (Deza and Vanstone (2010), Dukes and Sawchuck (2010), Frankl and Deza (1977), Chu *et al* (2004)) as well as exact and heuristic search techniques (Smith *et al* (2012), Barta *et al* (2014), Montemanni *et al* (2014a, 2014b), Janiszczack *et al* (2015)).

Studies about the MPCP show that in general the optimal, or the best known, solutions feature an extremely symmetric configuration, comparable to the structure of a crystal. In fact, optimal codes are usually obtained by combining selected orbits of specific permutation groups having considerable symmetry properties. On the other hand, permutation codes based on a random packing procedure are generally far from the optimum. From this point of view the MPCP shows remarkable similarities to the sphere packing problem in Euclidean space. The codewords of the MPCP correspond to the centers of spheres and the minimum Hamming distance constraint corresponds to the non-overlapping constraint of the spheres. One of the major mathematical achievements of the last decades was the proof of the Kepler Conjecture by Hales in 1998, stating that the face-centered cubic packing, that is the well-known regular way of piling cannonballs or oranges, is the tightest possible arrangement of spheres in space (see for instance Hales (2000)). As already proved by Gauss, this lattice packing has a density $\pi/\sqrt{18}$, which is about 74%. However, in recent years random sphere packings have been the object of systematic investigation and it has been proved in Song *et al* (2008) that random sphere packings cannot exceed a density limit of 63.4%.

The main purpose of this paper is to investigate, whether permutation codes behave similarly to sphere packings. More specifically, the regularity degree and the packing rate of many randomly generated permutation codes are measured and evaluated, in order to establish a relationship between these two variables. The paper starts with the formalization of the MPCP and with the definition of the regularity measures for permutation codes. The algorithm used for generating random permutation codes is then presented and finally the last section is devoted to the discussion of the computational results.

The maximum permutation code problem

Any *codeword* of length n in a permutation code can be obtained by permuting the n-tuple $x_0 = [0, 1, ..., n - 1] \in \mathbf{N}^n$. Let Ω_n be the set of all codewords of length n. A *permutation code C* is simply a set of codewords that is a subset of Ω_n. As already mentioned, the ability of permutation codes to identify and correct errors is related to the minimum Hamming distance between the elements. The Hamming distance $d_H(x, y)$ between codewords x and y is the number of components that differ in the two codewords. For any code $C \subseteq \Omega_n$ with $|C| > 1$ the *code distance* is defined as

$$\delta(C) = \min_{x,y \in C;\, x \neq y} d_H(x, y) \tag{1}$$

In other words, $\delta(C)$ corresponds to the minimum distance between the codewords belonging to the code C. The MPCP can now be formulated as follows.

Definition 1. Given a codeword length n and a distance d, the maximum permutation code problem MPCP consists of the determination of a largest code $C \subseteq \Omega_n$ that satisfies the *code distance constraint* $\delta(C) \geq d$.

In the sequel, MPCPs will be denoted by means of their characterizing parameters n and d and the maximum number of codewords of an (n, d)-problem will be indicated by $M(n,d)$. In analogy to the packing density of the sphere packing problem, the packing rate of a permutation code can be defined as follows:

Definition 2. Let C be a feasible code of an (n, d)-problem. The *packing rate* $\rho(C)$ is the ratio of the code size $|C|$ to the total number of permutations, that is

$$\rho(C) = \frac{|C|}{n!} \tag{2}$$

Measuring the packing regularity

When glass marbles are poured in a box, usually the random disposal of the spheres presents holes in-between, although no more marbles can be inserted. Randomly generated permutation codes of an (n, d)-problem have very similar features.

Definition 3. Let C be a permutation code of an (n, d)-problem. C is called a *maximal* code, if $\nexists C' \supset C$, such that $\delta(C') \geq d$.

In other words, no further codeword can be added to the code C without violating the code distance constraint. A well- known characteristic of the sphere packing problem is that the tightest possible arrangement of spheres has a highly regular pattern. An interesting issue is whether optimal permutation codes have analogous regularity features.

The distance pattern

The problem that arises is, how to measure the regularity degree of a feasible solution C of an (n, d)-problem. A natural approach is to look for regular patterns in the distance matrix of the solution.

Definition 4. Let $C = \{x_1, ..., x_{|C|}\}$ be a feasible code of an (n, d)-problem made up of $|C|$ codewords. Define the *distance matrix D* of the code C by $D(i,j) = d_H(x_i,x_j)$, $\forall i, j \in \{1, ..., |C|\}$.

The following definition introduces the concept of distance pattern of a codeword in a permutation code.

Definition 5. Let $f(k)$ be the number of occurrences of the distance $k \in \{d, ..., n\}$ between a codeword $x \in C$ and the rest of the codewords in the code C. We call the array $f = [f(d), ..., f(n)]$ the *distance pattern* of codeword x.

As the distance pattern provides an insight into the relative position of a codeword with respect to the other ones, the number of different distance patterns encountered in a code C, denoted by $\varphi(C)$, might be a simple but effective regularity indicator: the less different distance patterns are there, the more regular is the solution.

The sectorial balance

If the distance pattern gives information about the local structure around the codewords, the second regularity parameter that we adopt measures the overall homogeneity of the code within the complete set of codewords Ω_n. The idea of partitioning the search space Ω_n, introduced in Barta *et al* (2014) and applied in Barta *et al* (2015), Montemanni *et al* (2014a, 2014b), can be generalized as follows:

Definition 6. Denote by S_{ij} the set of codewords having the i-th component equal to j-1, that is $S_{ij} = \{x \in \Omega_n \mid x(i) = j-1\}$, $\forall\ i, j \in \{1, ..., n\}$.

It is interesting to remark that, by fixing the component index i, the collections of sets $\{S_{i1}, ..., S_{in}\}$ form partitions of Ω_n. However, also the collections of sets $\{S_{1j}, ..., S_{nj}\}$, obtained by fixing the value index j, form partitions of Ω_n. An effective way to measure the homogeneity of the distribution of a permutation code C, is to count the number of codewords belonging to each subset S_{ij}.

Definition 7. Let $P(i, j)$ be the number of codewords in a code C belonging to the sector S_{ij}. We refer to the $n \times n$ matrix P with components $P(i, j)$ as the *sectorial partition matrix* of the code C.

A peculiar property of the sectorial partition matrix P is that all its rows and columns sum up to $|C|$, since each of them represents a specific partition of the code. In a completely homogeneous solution it might be expected that the codewords are equally distributed with respect to the sectors, that is, each sector S contains exactly $\bar{p} = \frac{|C|}{n}$ codewords. In general, the balance degree of a code can be measured in the following manner.

Definition 8. Let P be the partition matrix of a code C. We define the *balance deviation dev*(C) of the code C by

$$dev(C) = \sqrt{\frac{1}{n^2}\sum_{i,j=1}^{n}(P(i,j) - \bar{p})^2} \tag{3}$$

The balance deviation $dev(C)$ corresponds to the standard deviation of the number of codewords in each sector S_{ij} with respect to the average value $\bar{p}$.

Optimal codes of problem (6,5)

The case of the problem (6,5) is particularly interesting, because it has a high number of optimal solutions, that is codes with size $|C| = 18$, however only four different structures have been observed. Table 1 summarizes the distance patterns of the 4 classes of optimal solutions $C_1, ..., C_4$. As shown in Table 1, the codes of type C_1 are perfectly regular, because all codewords have the same distance pattern and furthermore the balance deviation is equal to 0. This means that any C_1-solution has exactly 3 codewords in each sector S_{ij}. On the other hand, the classes C_2, C_3 and C_4 provide a remarkable example of optimal, but not completely symmetric permutation codes. For instance, codes of type C_2 have four different distance patterns (the number of their occurrence is reported in column 3) and a balance deviation equal to 0.408. Finally, the comparison of the values of $\varphi(C)$ and $dev(C)$ suggests a positive correlation between the two regularity indicators.

Generation of random permutation codes

Exact vs. heuristic algorithms

Theoretically, any MPCP can be solved by applying a suitable exact algorithm, such as a linear program or an exhaustive branch and bound search. However, as shown by several studies (see for instance Barta *et al* (2014), Bogaerts (2010), Montemanni *et al* (2015), Smith and Montemanni (2012)), exact algorithms can handle only small-sized (*n*,*d*)-problems, because of the combinatorial explosion of the search space Ω_n. Currently, state-of-the-art exact algorithms are able to solve to optimality only (*n*,*d*)-problems with $n \leq 6$ and some special cases of larger instances.

Table 1. Regularity degree of optimal (6,5)-codes

Code	Distance patterns		Codewords	$\varphi(C)$	$dev(C)$
	$f(5)$	$f(6)$			
C_1	12	5	18	1	0
C_2	15	2	1	4	0.408
	13	4	6		
	12	5	8		
	11	6	3		
C_3	15	2	1	4	0.577
	13	4	12		
	12	5	4		
	9	8	1		
C_4	15	2	1	5	0.577
	14	3	2		
	13	4	8		
	12	5	6		
	9	8	1		

On the other hand, many attempts have been made to generate feasible codes for larger instances by exploiting group theoretical knowledge on permutations (see Barta *et al* (2015), Chu *et al* (2004), Deza and Vanstone (1978)). The codes obtained by such algebraic approaches are obviously highly symmetric, but usually there is no way to tell whether they are optimal and whether there are other less regular equivalent solutions, as in the case of problem (6,5).

One main purpose of this study is to describe and test a fast heuristic algorithm, able to produce, by means of a random exploration of the search space, a large number of maximal codes for problems with $n > 6$.

Description of the algorithm CodeExplorer

The search algorithm, referred in the sequel as CodeExplorer, starts by generating an initial solution C by choosing codewords in a random way from the set of feasible codewords, denoted by *Rem*, which is initially set to Ω_n. Whenever a codeword x is picked from *Rem* and added to the code C, the whole neighbourhood $U(x) = \{x' \in Rem \mid d_H(x, x') < d\}$ of x is removed from *Rem*. This procedure is then repeated until *Rem* is empty.

At this point an exploration step is carried out, by retracting a given number N_{del} of the elements in C. The codewords to be deleted are chosen randomly. Consequently, the set *Rem* of the unused feasible codewords is updated and a new maximal code C_{new} is obtained by adding randomly chosen codewords from *Rem* until this set is exhausted again. If C_{new} represents an improvement in terms of the number of codewords, it is stored as the best current code. Finally, C_{new} is assigned to the current code C and a new exploration step can be performed. A more schematic description of the algorithm CodeExplorer is provided by the following pseudocode.

```
Step 1. Initialization
        Set $Rem := \Omega_n$, $C = \emptyset$ and $C_{best} = \emptyset$.
Step 2. Completion to a maximal code
```

Choose a random codeword $x \in Rem$. Update $C := C \cup \{x\}$ and $Rem := Rem \setminus U(x)$. Repeat Step 2 until $Rem = \emptyset$.

Step 3. *Update and exit criterion*

If $|C| > |C_{best}|$ update $C_{best} := C$. Stop if the maximum number of iterations is reached.

Step 4. *Retracting*

Select randomly N_{del} codewords out of C and remove them from C. Add to *Rem* all codewords $x \in \Omega_n \setminus C$ that are compatible with C (in terms of Hamming distance). Go to Step 2.

Computational experiments

The main purpose of the computational experiments reported in this section is to measure the regularity degree of a large number of maximal codes and to establish a relationship with the size of the solutions. The algorithm CodeExplorer described in the previous section is tailored for this task. At each iteration, the heuristic algorithm CodeExplorer builds a maximal code C of the (n,d)-problem and computes its regularity indicators, as explained before. In order to investigate the correlation between regularity and size, the values of the regularity indicators have been clustered depending on the code size $|C|$.

Tables 2-4 give a statistical overview of the regularity features of three different (n,d)-problems: (6,4), (6,5) and (7,5). For each instance 1,000,000 maximal codes have been generated and evaluated by our algorithm. The first column of each table shows the clusters of maximal codes ordered by code size. The following six columns report the minimum, the maximum, respectively the average value of the regularity indicators $\varphi(C)$, i.e. the number of different distance patterns within a code and $dev(C)$, the balance deviation, as previously defined. All the tests reported in this section have been obtained by running the algorithm CodeExplorer encoded in ANSI C on a computer equipped with an Intel Core *i*5 2.3 GHz processor and 8 GB of memory.

Since the optimum of problem (6,4) is known to be 120 (see for instance Smith and Montemanni (2012)), clusters of 10 units have been adopted in Table 2. The rate of retracted codewords at each iteration has been fixed to 60%. A higher rate would require significantly more computation time, since each step would involve an almost complete reconstruction of the code. Essentially, the low values of the regularity indicators $\varphi(C)$ and $dev(C)$ in the upper clusters of problem (6,4) show that, these solutions have a high degree of regularity. It is worth noticing that all optimal (6,4)-solutions found correspond to a unique, perfectly regular structure. As the code size decreases, the values of the regularity indicators clearly tend to increase, with a maximum around 70-80 codewords. In other words, a middle layer of mainly irregular codes can be observed between 50 and 90 codewords. Finally, the lowest clusters show a clear decrease in the regularity indicators. This effect is actually reasonable, because the codewords in extremely small maximal codes have to be placed so that no other codeword can fit in the empty spaces. Therefore it is not surprising to find high regularity in the low region.

Table 2. Size vs regularity: (6,4)-maximal codes

$\|C\|$	$\varphi(C)$			$dev(C)$		
	min	max	avg	min	max	avg
31-40	10	31	20.12	0.44	1.60	0.93
41-50	11	37	22.27	0.44	2.00	1.01
51-60	19	39	29.53	0.76	2.14	1.41
61-70	28	42	34.46	1.43	2.29	1.88
71-80	26	43	35.44	1.36	2.56	2.05
81-90	18	41	34.42	1.39	2.60	2.08
91-100	8	41	22.75	1.00	2.61	1.98
101-110	9	28	10.96	1.41	1.97	1.54
111-120	1	1	1.00	0.00	0.00	0.00

Table 3. Size vs regularity: (6,5)-maximal codes

$\|C\|$	$\varphi(C)$			$dev(C)$		
	min	max	avg	min	max	avg
6	1	1	1.00	0.00	0.00	0.00
7	2	5	3.54	0.37	0.37	0.37
8	3	6	4.72	0.47	0.53	0.48
9	2	7	4.72	0.50	0.69	0.52
10	2	8	4.62	0.47	0.78	0.54
11	1	8	4.62	0.37	0.90	0.55
12	1	8	4.73	0.00	0.94	0.55
13	2	8	4.95	0.37	0.99	0.57
14	2	8	5.17	0.47	0.97	0.59
15	1	8	5.37	0.50	0.99	0.61
16	1	7	5.42	0.47	1.05	0.62
17	3	7	5.24	0.37	0.90	0.62
18	1	5	2.94	0.00	0.58	0.29

Table 4. Size vs regularity: (7,5)-maximal codes

$\|C\|$	$\varphi(C)$			$dev(C)$		
	min	max	avg	min	max	avg
41-45	14	26	20.31	0.61	1.19	0.87
46-50	13	31	21.63	0.45	1.44	0.93
51-55	13	32	22.32	0.49	1.43	0.95
56-60	16	35	23.46	0.57	1.41	0.99
61-65	17	32	24.50	0.57	1.44	1.00
66-70	13	30	22.13	0.53	1.28	0.86
71-75	6	26	14.01	0.45	0.97	0.56
76-77	2	2	2.00	0.00	0.00	0.00

Table 5. Average code size and packing rate

n	d	$\|C\|$	packing rate (%)	
		avg	avg	best
6	4	40.79	5.67	16.67
6	5	12.03	1.67	2.50
7	5	49.80	0.99	1.53

As already observed in Table 1, problem (6,5) features four classes of optimal codes, each one of 18 codewords. A series of experiments with the algorithm CodeExplorer showed that about 42% of them are of the fully regular C_1-type, whereas about 58% of the optimal codes are slightly irregular! The results in Table 3 show highly regular patterns for $|C| = 18$, then a large central zone (between 8 and 17 codewords), containing a mix of mainly irregular structures and some regular codes. A higher degree of regularity can be observed again in the lowest clusters $|C| = 7$ and $|C| = 6$.

Table 4 contains the statistical results of problem (7,5). The fact that (7,5) is currently an open problem, that is not yet solved to optimality, makes it particularly challenging. Explicitly computed (7,5)-codes of size 77 can be obtained by assembling 11 orbits of a C_7 permutation group (for more details see Barta *et al* (2015), Smith and Montemanni (2012)). Due to the considerable size of the problem, it is not possible to generate in a reasonable time large solutions by adding codewords in a random way. Therefore, as an initial solution for the search with CodeExplorer we adopted a regular 77-code based on C_7-orbits with a retracting rate of 15%. The results presented in Table 4 are clustered with a cluster size of five units. The regularity profile of problem (7,5) presents strong similarities with the previously discussed instances: a peak of irregularity can be observed around 60 codewords and from these values upwards there is a clear decreasing trend. The largest solutions are 77-codes with a unique highly regular, fully balanced structure. However, these solutions are formed by two different distance patterns. A decreasing tendency of the regularity indicators is observable also downwards.

Finally, Table 5 compares the average packing rate of randomly generated codes with the packing rates of the best known solutions for the three (*n,d*)-problems considered. The retracting rate has been set to 100%. The results clearly show that on average random packings are significantly weaker than the best known codes.

Conclusions

This study focuses entirely on the metric structure of permutation codes and on a possible relationship between size and regularity. The results strengthen the intuition that large-sized and in particular optimal codes are in general also highly symmetric. However, a remarkable case of not completely regular but optimal code has been observed. The tools developed in this work might be helpful in the future to estimate, whether the current best solution of an open problem is likely to be the optimum or not.

References

Barta J., Montemanni R. and Smith D.H. (2014). A branch and bound approach to permutation codes. Proceedings of IEEE ICOICT, 187–192.

Barta J., Montemanni R. and Smith D.H. (2015). Permutation Codes via Fragmentation of Group Orbits, Proceedings of IEEE ICOICT, 39–44.

Bogaerts M. (2010). New upper bounds for the size of permutation codes via linear programming. The El. Jour. of Combinatorics. 17(#R135).

Chu W., Colbourn C.J. and Dukes P. (2004). Constructions for permutation codes in powerline communications. Designs, Codes and Cryptography 32, 51–64.

Colbourn C.J., Kløve T. and Ling A.C.H. (2004). Permutation arrays for powerline communication and mutually orthogonal latin squares. IEEE Trans. Inform. Theory 50, 1289–1291.

De la Torre D.R., Colbourn C.J. and Ling A.C.H. (2000). An application of permutation arrays to block ciphers, Proceedings of the 31st Int. Conf. on Combinatorics, Graph theory and Computing vol 145, 5-7.

Deza M. and Vanstone S.A. (1978). Bounds for permutation arrays. J. Statist. Plann. Inference 2, 197–209.

Dukes P. and Sawchuck N. (2010). Bounds on permutation codes of distance four. J. Alg. Comb. 31, 143–158.

Frankl P. and Deza M. (1977). On maximal numbers of permutations with given maximal or minimal distance. J. Combin. Theory Ser. A 22, 352–260.

Hales T. (2000). Cannonballs and honeycombs. Notices of the American Mathematical Society. 47(4), 440–449.

Han Vinck A.J. (2000). Coded modulation for power line communications. A.E.U. Int. J. Electron. Commun. 54(1), 45–49.

Janiszczak I., Lempken W., Ostergard P.R.J. and Staszewski R. (2015) Permutation codes invariant under isometries. Designs, Codes and Cryptography 75(3), 497-507.

Jiang A., Mateescu R., Schwartz M. and Bruck J., Rank modulation for flash memories. Proceedings of the IEEE Symposium on Information Theory, 1731-1735, 2008.

Montemanni R., Barta J. and Smith D.H. (2014a). Permutation codes: a branch and bound approach. Proceedings of PMAMCM, 86–90.

Montemanni R., Barta J. and Smith D.H. (2014b). Permutation codes: a new upper bound for M(7,5). Proceedings of ICIAC, 1–3.

Montemanni R., Barta J. and Smith D.H. (2015). The design of permutation codes via a specialized maximum clique algorithm. Proceedings of IEEE MCSI.

Pavlidou N., Han Vinck A. J., Yazdani J. and Honary B. (2003). Powerline communications: state of the art and future trends. IEEE Communications Magazine 41(4), 34–40.

Smith D.H. and Montemanni R. (2012). A new table of permutation codes. Design, Codes and Cryptography 63(2), 241–253.

Song C., Wang P. and Makse H.A. (2008). A phase diagram for jammed matter. Nature 453 (7195), 629–632.

Tarnanen H. (1999). Upper bounds on permutation codes via linear programming. Eur. J. Combin. 20, 101–114.