Des. Codes Cryptogr. (2012) 63:241–253
DOI 10.1007/s10623-011-9551-8

A new table of permutation codes

Derek H. Smith · Roberto Montemanni

Received: 24 March 2011 / Revised: 21 July 2011 / Accepted: 23 July 2011 /
Published online: 19 August 2011

Abstract Permutation codes (or permutation arrays) have received considerable interest in recent years, partly motivated by a potential application to powerline communication. Powerline communication is the transmission of data over the electricity distribution system. This environment is rather hostile to communication and the requirements are such that permutation codes may be suitable. The problem addressed in this study is the construction of permutation codes with a specified length and minimum Hamming distance, and with as many codewords (permutations) as possible. A number of techniques are used including construction by automorphism group and several variations of clique search based on vertex degrees. Many significant improvements are obtained to the size of the best known codes.

Keywords Permutation codes · Permutation arrays · Automorphism groups · Clique search

Mathematics Subject Classification (2000) 05A05 · 05E20 · 94B60

1 Introduction

Permutation codes (sometimes called permutation arrays) have been proposed in [12] for use with a specific modulation scheme. An account of the rationale for the choice of permutation codes can be found in [7] (see also [13,21]). Permutations are used to ensure that power output remains as constant as possible. As well as white Gaussian noise the codes

Communicated by C. J. Colbourn.

D. H. Smith
Division of Mathematics and Statistics, University of Glamorgan, Pontypridd, CF37 1DL, Wales, UK
e-mail: dhsmith@glam.ac.uk

R. Montemanni (✉)
Istituto Dalle Molle di Studi sull'Intelligenza Artificiale (IDSIA), Galleria 2, 6928, Manno, Switzerland
e-mail: roberto@idsia.ch

must combat permanent narrow band noise from electrical equipment or magnetic fields, and impulsive noise.

A permutation code is simply a set of permutations in the symmetric group $\mathscr{S}_n$ of all permutations on n elements. The codewords are the permutations and the code length is n. The ability of a permutation code to correct errors is related to the *minimum Hamming distance* of the code. The Hamming distance δ between two codewords is the number of elements that differ in the two permutations. Alternatively, two permutations σ_1 and σ_2 are at distance δ if $\sigma_1\sigma_2^{-1}$ has exactly $n-\delta$ fixed points. The minimum distance d is then the minimum δ taken over all pairs of distinct permutations. A code of length n with minimum distance d will be denoted an (n, d) permutation code and the maximum number of codewords in such a code is denoted by $M(n, d)$.

A central question in the theory of permutation codes is the determination of $M(n, d)$, or of good lower bounds for $M(n, d)$. The most complete contribution to this question is in [7]. Other contributions are indicated in detail in the tables and in Sect. 6. Upper bounds for $M(n, d)$ are also known and the upper and lower bounds can coincide when d is close to n.

The new table of values of lower bounds for $M(n, d)$ with a significant number of improved values is presented in Sect. 2. A variety of maximum clique algorithms used in this study are described in Sect. 3. The automorphism groups used in many of the cases are detailed in Sect. 4. Iterative clique building methods used in many of the other cases and based on either codewords, cycles of length n or cycles of length $n-1$ are described in Sect. 5. Methods used in other cases are given in Sect. 6. The conclusion summarizes the success of the various methods.

2 The new table

Cases with $d = 2$ and $d = 3$ correspond to the permutations of the symmetric group $\mathscr{S}_n$ (giving $M(n, 2) = n!$) and the alternating group $\mathscr{A}_n$ (giving $M(n, 3) = n!/2$) respectively, and so the table only need include values of d with $4 \le d \le n$. The result $M(n, n) = n$ arises from a single orbit of a cyclic group $\mathscr{C}_n$. If q is a prime power then Frankl and Deza [11] show that $M(q, q-1) = q(q-1)$, $M(q+1, q-1) = (q+1)q(q-1)$. These results arise from the fact that $AGL(1, q)$ is sharply 2-transitive and $PGL(2, q)$ is sharply 3-transitive (see also [2] and Proposition 1.2 of [7]). None of these standard results will be specially marked in the table. As standard cases cover $n = 5$ completely, the table will address the cases $6 \le n \le 18$.

The key to the methods used and to the sources of upper bounds and previously known lower bounds is given in Table 1. The table of permutation codes itself is given in Table 2 for $4 \le d \le 8$ and in Table 3 for $9 \le d \le 18$. An entry in bold is a new best lower bound, and an entry in italics equals the previous best lower bound. Two other new results, for $M(19, 17)$ and $M(20, 19)$, are outside the limits of the table and are given in Sect. 5.

3 Maximum clique algorithms

The use of clique search to find (n, d) permutation codes is described in [7]. In the simplest application of clique search a graph $G(n, d)$ is built with vertices corresponding to permutations in $\mathscr{S}_n$, and two vertices are adjacent if the two permutations have Hamming distance

Table 1 Key to the superscripts used in Tables 2 and 3

Unmarked entries LB	Obtained using an automorphism group with clique search
Superscript A	"Iterative clique building"—codewords only (Sect. 5)
Superscript C	"Iterative clique building"—cycles of length n (Sect. 5)
Superscript E	"Iterative clique building"—cycles of length $n-1$ (Sect. 5)
Superscript X	Other method (Sect. 6)
Unmarked entries UB or old LB	Taken from [11]
Superscript a	Taken from [7]
Superscript b	Taken from [8]
Superscript c	Taken from [9]
Superscript d	Taken from [16]
Superscript e	Upper bound taken from [22]
Superscript f	Taken from [10]
Superscript g	Upper bound taken from Theorem 3 of [11]
Superscript h	Upper bound taken from [3]

at least d. The size of the largest clique (complete subgraph) in $G(n, d)$ gives the value of $M(n, d)$. When an automorphism group is used, clique search can be applied to a graph with vertices representing the orbits of the automorphism group. Two vertices are adjacent if the distance between the orbits is at least d. The maximum clique then gives a union of orbits forming a code. The size of this code is a lower bound for $M(n, d)$.

In either case any efficient algorithm can be used if the clique search terminates so that the largest clique is obtained. However, the maximum clique problem is NP-complete. If the problem is too large for the algorithm to terminate a variety of heuristic approaches are available, and these are further developed in this work. It can be helpful to try several approaches. Algorithms that allow vertex orderings based on vertex degrees have some advantages for these problems and will be described first.

The software system FASoft used for radio frequency assignment [15] contains a maximum clique algorithm based on that described in [6]. The effectiveness of this algorithm can depend strongly on the order in which the vertices of the graph are presented. This is true both for the speed of termination of the algorithm, and the quality of the solution available if the algorithm does not terminate. The system allows thirteen different vertex orderings, based on vertex degrees and edge weights. As the graph has unweighted edges, only seven of the thirteen orderings are distinct here. These are listed in [15] and the associated software documentation as:

Initial ordering: The algorithm in [6] is applied with the order of vertices as presented by the problem.
Largest degree first (LF1): The vertices are sorted in decreasing order of their degrees before the algorithm is applied.

LF1 reversed: The reverse of the above ordering.
Largest degree first (LF2): The vertices of largest degree are successively removed from the graph and added to a list. This time the degree calculation excludes vertices that have already been ordered and removed from the graph.
LF2 reversed: The reverse of the above ordering.
Smallest degree last (SL): The vertices of smallest degree are successively removed from the graph and added to a list. Again the degree calculation excludes vertices that have already been ordered and removed from the graph. When all vertices have been removed the list is reversed.
SL reversed: The reverse of the above ordering.

These seven orderings were applied to graphs with vertices corresponding to orbits of automorphism groups. When the algorithm did not terminate it was found that very different results could be obtained from the different orderings in any reasonable run time. It was found best to run all seven orderings quickly and then select the ordering that gave the largest clique. The algorithm was then run with that ordering for as long as was practical.

Some variations on the orderings were also considered within a multi-start framework in which the algorithm described in [6] was repeatedly run for a short period (600 s), each time with perturbations of an original ordering. This ordering was selected after tests with the orderings in FASoft. This use of perturbed vertex orderings led to improved results for some of the best automorphism groups, and also for some sub-optimal automorphism groups. Specifically, once the most promising ordering has been selected, at each iteration the ordering is perturbed by introducing some noise in the calculation of the degree of each node. Given deg_i the actual degree of node i, it is changed into a random value in the interval $[(1 - Per)deg_i, (1 + Per)deg_i]$, where Per is a user-defined parameter. For these experiments values of Per in the interval $[0.05, 0.15]$ were used to regulate the magnitude of the perturbation. Ideally, the perturbed ordering should not be very different from the most promising ordering, but sufficiently different to produce different solutions.

Some other algorithms, both heuristic and (truncated) exact, were also considered for solving the maximum clique problem. In particular, some experiments used the methods discussed in [4,19,20]. Sometimes using these methods led to better results than the use of the various degree orderings available with FASoft, or the use of perturbed orderings. The algorithm described in [17] was also tested, but did not give the best results for any of the problems that arose in this study.

4 Automorphism groups

An (n, d) permutation code Γ has an automorphism group H if $h\Gamma = \Gamma$ for all $h \in H$. If H is applied to the identity permutation a single orbit is obtained. In general a code Γ with this automorphism group will consist of a union of $|\Gamma|/|H|$ orbits. Clearly it is necessary to choose the automorphism group so that the minimum distance between any pair of codewords in a single orbit (referred to here as the *internal minimum distance*) is at least d. It is sometimes convenient to choose a single permutation as a representative of each orbit. In this work the lexicographically minimal permutation has been chosen. The graph $G(n, d)$ has one vertex for each orbit of H. Given two orbits O_i and O_j corresponding to vertices v_i and v_j of $G(n, d)$, define the distance between the orbits as $\delta(O_i, O_j) = \min\ \delta(g_i, g_j)$ where the minimum is taken over all $g_i \in O_i$ and all $g_j \in O_j$. Then two vertices v_i and v_j of $G(n, d)$ are adjacent if $\delta(O_i, O_j) \geq d$. A maximum clique in $G(n, d)$ then corresponds to

the largest (n, d) code with the given automorphism group H. Of course there may be larger codes with a different automorphism group.

In [7] the automorphism groups used were the cyclic group $\mathscr{C}_n$, the dihedral group $\mathscr{D}_{2n}$, the groups $AGL(1, q)$, $PGL(2, q)$ and the Mathieu groups M_{11} and M_{12}. In the present work this selection was extended to include $\mathscr{C}_{n-1}$, $A\Gamma L(1, q)$, $A\Sigma L(1, q)$, $P\Gamma L(2, q)$, $P\Sigma L(2, q)$, and some of the largest groups from a database of transitive groups. All computations using automorphism groups were carried out using Magma[1]. Magma contains a database of all transitive groups with degree at most 30. This is based on one constructed by Hulpke [14] making use of a classification by Butler and McKay for degree at most 15. Usually the largest two or three groups with internal minimum distance at least d were considered. Outlines of the constructions of all these groups, details of the relevant Magma functions and information on the database can be found in [5]. It was not always the largest group that gave the largest permutation code, but large groups are certainly good candidates. A more efficient use of $\mathscr{C}_n$ and $\mathscr{C}_{n-1}$ will be considered in Sect. 5.

Automorphism groups used in individual cases in the tables will now be listed. Here all permutations act on the set $\{0, 1, 2, \ldots, n-1\}$. Standard cases mentioned in Sect. 2 are omitted unless an alternative construction is used. Cases with a single orbit (which include these standard cases) are group codes [1] and the decoding algorithm given in [1] could be used for these codes. Permutation generators for the automorphism groups are given. Files of codewords and files of orbit representatives which also list these permutation generators can be found on the authors' web pages[2].

1. $(n,d)=(8, 4)$: Use $PSL(2, 7)$ with $|PSL(2, 7)| = 168$ and internal orbit minimum distance 8. A clique search algorithm terminated with 16 orbits, so $M(8, 4) \geq 2688$. Permutation generators used were: (2 5 6)(3 4 7); (0 5 1)(2 7 6).
2. $(n,d)=(8,5)$: Use $AGL(1, 8)$ with $|AGL(1, 8)| = 56$ and internal orbit minimum distance 7. A clique search algorithm terminated with 11 orbits, so $M(8, 5) \geq 616$. Permutation generators used were: (0 1 2 3 4 5 6); (0 7)(1 3)(2 6)(4 5).
3. $(n,d)=(8,7)$: Use $\mathscr{C}_7$, with $|\mathscr{C}_7| = 7$ and internal orbit minimum distance 7. A clique search algorithm terminated with 8 orbits, so referring to the upper bound $M(8, 7) = 56$.
4. $(n,d)=(9,4)$: Use $P\Gamma L(2, 8)$ with $|P\Gamma L(2, 8)| = 1512$ and internal orbit minimum distance 6. A clique search algorithm terminated with 12 orbits, so $M(9, 4) \geq 18144$. Permutation generators used were:
   (2 8 4 7 5 3 6); (0 2 1)(3 6 7)(4 5 8); (3 7 6)(4 5 8).
5. $(n,d)=(9,5)$: Use $A\Gamma L(1, 9)$ with $|A\Gamma L(1, 9)| = 144$ and internal orbit minimum distance 6. A clique search algorithm terminated with 21 orbits, so $M(9, 5) \geq 3024$. In fact the FASoft algorithm did not terminate in 1–2 days whereas the algorithm described in [19,20] terminated in 9200 s on a 2.4GHz machine. Permutation generators used were:
   (0 1 2 3 4 5 6 7); (0 4 8)(1 2 7)(3 6 5); (1 3)(2 6)(5 7).
6. $(n,d)=(9,6)$: Use $PGL(2, 8)$ with $|PGL(2, 8)| = 504$ and internal orbit minimum distance 7. A clique search algorithm terminated with 3 orbits, so $M(9, 6) \geq 1512$. Permutation generators used were: (2 8 4 7 5 3 6); (0 2 1)(3 6 7)(4 5 8).
7. (n,d): Use $P\Sigma L(2, 9)$ with $|P\Sigma L(2, 9)| = 720$ and internal orbit minimum distance 6. The clique search algorithm in [4] found 209 orbits without terminating, so

[1] http://magma.maths.usyd.edu.au/magma/.

[2] http://data.research.glam.ac.uk/projects/; http://www.idsia.ch/~roberto/permutationcodes11.zip.

$M(10, 4) \geq 150480$. Permutation generators used were:
(2 8 6 7)(3 9 4 5); (0 8 1)(2 9 5)(3 6 4); (2 6)(3 5)(4 9).

8. (*n*,*d*)=(10,5): Use $P\Sigma L(2, 9)$ as for $(n, d) = (10, 4)$. The use of perturbed vertex orderings in a multi-start clique search algorithm found 26 orbits without terminating, so $M(10, 5) \geq 18720$.
9. (*n*,*d*)=(10,6): Use $PSL(2, 9)$ with $|PSL(2, 9)| = 360$ and internal orbit minimum distance 8. A clique search algorithm terminated with 24 orbits, so $M(10, 6) \geq 8640$. Permutation generators used were: (2 8 6 7)(3 9 4 5); (0 8 1)(2 9 5)(3 6 4).
10. (*n*,*d*)=(10,7): Use $PGL(2, 9)$ with $|PGL(2, 9)| = 720$ and internal orbit minimum distance 8. Only a single orbit is possible, so $M(10, 7) \geq 720$. Permutation generators used were: (2 4 8 5 6 3 7 9); (0 8 1)(2 9 5)(3 6 4).
11. (*n*,*d*): Use the Mathieu group M_{11} with $|M_{11}| = 7920$ and internal orbit minimum distance 8. The clique search algorithm in [4] found 220 orbits without terminating, so $M(11, 4) \geq 1742400$. Permutation generators used were:
(0 1 2 3 4 5 6 7 8 9 10); (0 2 8 4 3)(1 5 6 9 7); (1 5 9 6)(2 8 3 4).
12. (*n*,*d*): Use the Mathieu group M_{11} as for $(n, d) = (11, 4)$. The use of perturbed vertex orderings in a multi-start clique search algorithm found 26 orbits without terminating, so $M(11, 5) \geq 205920$.
13. (*n*,*d*)=(11,6): Use the Mathieu group M_{11} as for $(n, d) = (11, 4)$. A clique search algorithm terminated with 12 orbits, so $M(11, 6) \geq 95040$.
14. (*n*,*d*): Use the Mathieu group M_{11} as for $(n, d) = (11, 4)$. Only a single orbit is possible, so $M(11, 7) \geq 7920$.
15. (*n*,*d*)=(11,8): Use the Mathieu group M_{11} as for $(n, d) = (11, 4)$. Only a single orbit is possible, so referring to the upper bound $M(11, 8) = 7920$.
16. (*n*,*d*)=(11,10): Use $\mathscr{C}_{11}$, with $|\mathscr{C}_{11}| = 11$ and internal orbit minimum distance 11. A clique search algorithm terminated with 10 orbits, so referring to the upper bound $M(11, 10) = 110$.
17. (*n*,*d*)=(12,4): Use the Mathieu group M_{12} with $|M_{12}| = 95040$ and internal orbit minimum distance 8. The clique search algorithm in [4] found 220 orbits without terminating, so $M(12, 4) \geq 20908800$. Permutation generators used were:
(0 10 1 2 3)(4 7 11 5 6); (0 8 4 11 10 7 1 3)(5 9).
18. (*n*,*d*)=(12,5): Use the Mathieu group M_{12} as for $(n, d) = (12, 4)$. The use of perturbed vertex orderings in a multi-start clique search algorithm found 25 orbits without terminating, so $M(12, 4) \geq 2376000$.
19. (*n*,*d*)=(12,6): Use the Mathieu group M_{12} as for $(n, d) = (12, 4)$. A clique search algorithm terminated with 2 orbits, so $M(12, 6) \geq 190080$.
20. (*n*,*d*): Use the Mathieu group M_{12} as for $(n, d) = (12, 4)$. Only a single orbit is possible, so $M(12, 7) \geq 95040$.
21. (*n*,*d*)=(12,8): Use the Mathieu group M_{12} as for $(n, d) = (12, 4)$. Only a single orbit is possible, so referring to the upper bound $M(12, 8) = 95040$.
22. (*n*,*d*)=(12,9): Use $PGL(2, 11)$ with $|PGL(2, 11)| = 1320$ and internal orbit minimum distance 10. Only a single orbit is possible, so $M(12, 9) \geq 1320$. Permutation generators used were:
(0 1 2 3 4 5 6 7 8 9 10); (0 9)(1 4)(2 6)(3 7)(5 8)(10 11); (0 1 3 7 4 9 8 6 2 5).
23. (*n*,*d*)=(12,11): Use $\mathscr{C}_2 \times \mathscr{C}_6$ with $|\mathscr{C}_2 \times \mathscr{C}_6| = 12$ and internal orbit minimum distance 12. A clique search algorithm terminated with 5 orbits, so $M(12, 11) \geq 60$. In fact this code comes from 5 mutually orthogonal latin squares [8]. Permutation generators used were:
(0 7 2 9 4 11)(1 8 3 10 5 6); (0 6)(1 7)(2 8)(3 9)(4 10)(5 11).

24. (n,d)=(14,10): Use $PGL(2, 13)$ with $|PGL(2, 13)| = 2184$ and internal orbit minimum distance 12. A clique search algorithm terminated with 3 orbits, so $M(14, 10) \geq 6552$. Permutation generators used were: (2 8 9 11 5 12 3 10 4 6 7 13); (0 12 1)(2 4 8)(3 5 13)(6 9 7).
25. (n,d)=(14,11): Use $PGL(2, 13)$ as for $(n, d) = (14, 10)$. Only a single orbit is possible, so $M(14, 11) \geq 2184$.
26. (n,d)=(15,12): Use the transitive permutation group (15,47) in the Magma database with order 2520 and internal orbit minimum distance 12. Only a single orbit is possible, so $M(15, 12) \geq 2520$. Permutation generators used were: (0 8 9 2 13)(1 14 6 11 5)(3 4 10 12 7); (0 1 2)(4 5 6)(7 9 8)(11 13 12).
27. (n,d)=(16,12): Use the transitive permutation group (16,1840) in the Magma database with order 40320 and internal orbit minimum distance 12. Only a single orbit is possible, so $M(16, 12) \geq 40320$. Permutation generators used were: (0 5 4 6 1 3 15)(7 8 13 12 14 9 11); (0 8 2 13 11 10 3)(1 12 7 15 9 6 4).

Given the nature of the maximum clique algorithms, it is unlikely that a further orbit could be added to any of these codes. For codes with $n \leq 10$ attempts were made to add just a single codeword (trying all possible permutations). These experiments were unsuccessful. Thus it seems that the codes obtained are maximal with respect to inclusion.

5 Iterative clique building

For some of the permutation code instances, the underlying graph obtained as described in Sect. 3 is too large to handle directly by a maximum clique algorithm. This sometimes happens even when automorphisms are used to create the graph, as described in Sect. 4. For such problems a different algorithm, originally described in [18] for a different coding theory problem, is used. It is again based on maximum clique calculations, but in a way that leads to smaller maximum clique problems. The method is iterative, and starts from a given solution, that can be either provided by the user or generated by a lexicographic search method (see [18]), which is a fast greedy method.

Starting from a given initial solution, a random subset of the permutations of the code is removed, leaving a partial code. A parameter $CSRem$ defines the percentage of permutations removed. All codewords compatible with those left in the code can be identified via a lexicographic search, and a graph can be built among these feasible permutations, as described in Sect. 3. Notice that this graph is much smaller than the complete one. A maximum clique problem is then solved to complete the partial code. If the code obtained is larger than the starting one, it becomes the new reference solution, otherwise the starting one is kept. The procedure is iteratively repeated, always starting from the best solution available. The method stops when a given maximum computation time (a few days in our experiments) has elapsed. To solve the maximum clique problem the algorithm described in [6] was used, or alternatively its multi-start iterative modification described in Sect. 3. These internal maximum clique algorithms are run for a maximum time of 3600 s at each iteration. The parameter $CSRem$ took values between 8 and 16 in these experiments.

This iterative clique building approach has been applied as described above to graphs obtained directly from permutations (superscript A in Tables 2, 3). It has also been applied to graphs obtained using cyclic automorphisms $\mathscr{C}_n$ (superscript C) and $\mathscr{C}_{n-1}$ (superscript E). As the method is feasible for larger values of n, it has been further applied to give two new best results $M(19, 17) \geq 343$ (upper bound 5814) and $M(20, 19) \geq 78$ (upper bound

380) that are not recorded in the table. These were both obtained by direct application of the method to permutations.

It is possible to use codes generated with automorphism groups as initial solutions for iterative clique building. Although some success with this approach was obtained in one or two cases, these cases were not ones that appear in the final table. It seems unlikely that improvements could be obtained in this way for codes with a large automorphism group.

6 Other methods

There were just two cases in the range of parameters covered by Tables 2 and 3 where the methods described in Sects. 4 and 5 were unable to match the best known result:

For the case $(n,d)=(7,4)$ iterative clique building was only able to find a code with 348 codewords, where the best known code has 349 codewords. The use of an automorphism group only gave 336 codewords. A code with 349 codewords was obtained as follows. A random clique of $G(7,4)$ was generated greedily using a first random ordering of the permutations of $\mathscr{S}_7$. At iteration i, 10–12% of the vertices were deleted from the current clique and vertices representing permutations of $\mathscr{S}_7$ were added (together with with the edges of $G(7,4)$ induced by the vertex) according a new ith random ordering of $\mathscr{S}_7$. Vertices were only added provided the subgraph of $G(7,4)$ obtained was itself a clique. The method only seems suitable for small n.

For the case $(n,d)=(10,9)$ the methods described in Sects. 4 and 5 did not give a code with more than 36 codewords. A code with 49 codewords can be obtained by following precisely the method of Janiszczak and Staszewski [16]. Let U be the subgroup of $\mathscr{S}_{10}$ generated by the element (0 1 2 3 4 5 6 7)(8 9). Let B be the set of permutations $\{(0\,1\,9\,3\,6\,4\,2\,5\,7\,8);\ (0\,1\,2\,6\,8\,9\,7\,3\,5);\ (0\,3\,6\,2\,9\,7\,4)(1\,8);\ (0\,2\,7\,6\,9\,1\,8)(4\,5);\ (0\,1\,4\,8\,3\,9\,6)(2\,7);\ (0\,3\,8\,9)(2\,7\,5)(4\,6);\ \mathrm{id}\} \subset \mathscr{S}_{10}$. Then $\cup_{b\in B} b^U$ gives a code with 49 codewords.

7 Conclusion

Careful attention to the selection of automorphism groups in conjunction with a combination of maximum clique algorithms, together with the use of iterated clique building, has led to many new codes. Twenty four of these codes have more codewords than the best code previously known. The increase in the number of codewords is very substantial in most cases. The combination of methods has also matched the number of codewords of the best code previously known in the range of the tables, with special attention necessary in only three cases.

Files of codewords for all the codes presented (including those in Sect. 6) have been made available on the authors' web pages. These files have been fully checked except for the one case $(n,d)=(12,4)$ which is too large to check fully, but has been extensively checked.

The difficulty of some of the maximum clique problems generated is notable. Significantly different results can be obtained from different algorithms, or from runs of a single algorithm with different vertex orderings. Sometimes the size of maximum clique given by long but non-terminated runs of different algorithms can differ by a factor of two or more. Thus the problems may prove useful challenges for future research on maximum clique algorithms.

The methods become much harder to apply if n is increased further. Often it is not even possible to build the clique problem in reasonable time or to store it, and finding a good initial solution for iterated clique building becomes very time consuming. It is possible to

Table 2 The new table of permutation codes $4 \leq d \leq 8$

n\d		4	5	6	7	8
6	UB	120	18^{c}	6	–	–
	LB	*120*^{C}	*18*^{A}	*6*	–	–
	old LB	120	18^{c}	6	–	–
7	UB	543^{e}	140^{e}	42	7	–
	LB	*349*^{X}	*77*^{C}	*42*^{C}	*7*	–
	old LB	349^{a}	77^{a}	42	7	–
8	UB	4135^{e}	926^{e}	336	56	8
	LB	*2688*	**616**	*336*^{C}	*56*	*8*
	old LB	2688^{a}	560^{a}	336	56	8
9	UB	32989^{e}	7128^{e}	1962^{e}	504	72
	LB	*18144*	**3024**	*1512*	*504*^{C}	*72*^{E}
	old LB	18144^{a}	1944^{a}	1512^{a}	504	72
10	UB	302400^{e}	64800^{e}	16941^{e}	4699^{e}	720
	LB	**150480**	**18720**	**8640**	*720*	*720*
	old LB	86400^{a}	13680^{a}	4320^{a}	720	720
11	UB	3326400^{e}	362880^{h}	138600^{h}	32874^{h}	7920
	LB	**1742400**	**205920**	**95040**	*7920*	*7920*
	old LB	950400^{a}	60940^{a}	9790^{a}	7920^{a}	7920^{a}
12	UB	39916800^{e}	4354560^{h}	1663200^{h}	361396^{h}	95040
	LB	**20908800**	**2376000**	**190080**	*95040*	*95040*
	old LB	11404800^{a}	731280^{a}	117480^{a}	–	95040^{a}

Table 2 continued

n\d		4	5	6	7	8
13	UB	367270674[h]	56609280[h]	21621600[h]	4163390[h]	879493[h]
	LB	–	–	–	–	**27132**[E]
	old LB	41712480[a]	878778[a]	271908[a]	–	–
14	UB	6227020800[g]	792529920[h]	302702400[h]	58287460[h]	12312902[h]
	LB	–	–	–	–	–
	old LB	550368000[a]	–	–	–	–
15	UB	87178291200[g]	12336550641[g]	4540536000[g]	1287081070[g]	259459200
	LB	–	–	–	–	–
	old LB	7925299200[a]	–	–	–	–
16	UB	1162377216000[f]	172915618909[g]	63210845583[g]	16859621182[g]	3921062572[g]
	LB	–	–	–	–	–
	old LB	126804787200[a]	–	–	–	–
17	UB	18197961437469[f]	2596258599240[g]	943467979034[g]	237600152368[g]	54411416260[g]
	LB	–	–	–	–	–
	old LB	–	–	–	–	–
18	UB	320118685286400[f]	41573855232000[g]	15029046257577[g]	3584755714293[g]	809812004266[g]
	LB	–	–	–	–	–
	old LB	–	–	–	–	–

UB upper bound, *LB* lower bound given by a code constructed in this study. *oldLB* best lower bound available from a construction in the literature. An entry in *bold* is a new best lower bound, and an entry in *italics* equals the previous best lower bound

Table 3 The new table of permutation codes $9 \leq d \leq 18$

n\d		9	10	11	12	13	14	15	16	17	18
9	UB	9	–	–	–	–	–	–	–	–	–
	LB	*9*	–	–	–	–	–	–	–	–	–
	old LB	9	–	–	–	–	–	–	–	–	–
10	UB	90	10	–	–	–	–	–	–	–	–
	LB	*49*^{X}	*10*	–	–	–	–	–	–	–	–
	old LB	49^{d}	10	–	–	–	–	–	–	–	–
11	UB	990	110	11	–	–	–	–	–	–	–
	LB	*154*^{C}	*110*	*11*	–	–	–	–	–	–	–
	old LB	154^{a}	110	11	–	–	–	–	–	–	–
12	UB	11880	1320	132	12	–	–	–	–	–	–
	LB	*1320*	*1320*	*60*	*12*	–	–	–	–	–	–
	old LB	–	1320	60^{b}	12	–	–	–	–	–	–
13	UB	154440	17160	1716	156	13	–	–	–	–	–
	LB	**4810**^{A}	**906**^{A}	**195**^{C}	*156*	*13*	–	–	–	–	–
	old LB	3588^{a}	–	–	156	13	–	–	–	–	–
14	UB	2162160	240240	24024	2184	182	14	–	–	–	–
	LB	–	*6552*	*2184*	*2184*	**52**^{A}	*14*	–	–	–	–
	old LB	–	6552	–	2184	42^{b}	14	–	–	–	–
15	UB	32432400	3603600	360360	32760	2730	210	15	–	–	–
	LB	–	–	**6076**^{E}	**2520**	**243**^{A}	**56**^{A}	*15*	–	–	–
	old LB	–	–	–	–	84^{a}	–	15	–	–	–

Table 3 continued

n\d		9	10	11	12	13	14	15	16	17	18
16	UB	518918400	57657600	5765760	524160	43680	3360	240	16	–	–
	LB	–	–	–	**40320**	**1266**^{A}	**269**^{A}	*240*	*16*	–	–
	old LB	–	–	–	21120^{a}	–	–	240	16	–	–
17	UB	8821612800	980179200	98017920	8910720	742560	57120	4080	272	17	–
	LB	–	–	–	–	–	–	*4080*	*272*	*17*	–
	old LB	–	–	–	83504^{a}	–	–	4080	272	17	–
18	UB	158789030400	17643225600	1764322560	160392960	13366080	1028160	73440	4896	306	18
	LB	–	–	–	–	–	–	–	*4896*	**70**^{A}	*18*
	old LB	–	–	–	–	–	–	–	4896	54^{b}	18

UB upper bound, *LB* lower bound given by a code constructed in this study. *old LB* best lower bound available from a construction in the literature. An entry in *bold* is a new best lower bound, and an entry in *italics* equals the previous best lower bound

use various forms of incomplete greedy search, but there is no reason to think that the codes obtained will be particularly good in terms of the number of codewords. Inspection of the tables and the two results for $n = 19$ and $n = 20$ indicate that iterated clique building is particularly useful for larger n.

Acknowledgments The authors would like to thank Peter Dukes for suggesting parameter sets on which to concentrate their efforts.

References

1. Bailey R.F.: Error-correcting codes from permutation groups. Discrete Math. **309**, 4253–4265 (2009).
2. Blake I.F.: Permutation codes for discrete channels. IEEE Trans. Inform. Theory **20**(1), 138–140 (1974).
3. Bogaerts M.: New upper bounds for the size of permutation codes via linear programming. Electron. J. Combi. **17**(#R135) (2010).
4. Burer S., Monteiro R.D.C., Zhang Y.: Maximum stable set formulations and heuristics based on continuous optimization. Math. Progr. Ser A. **94**, 137–166 (2002).
5. Cannon J.J., Bosma W. (eds.): Handbook of magma functions, version 2.13. The University of Sydney, Sydney (2006).
6. Carraghan R., Pardalos P.M.: An exact algorithm for the maximum clique problem. Oper. Res. Lett. **9**, 375–382 (1990).
7. Chu W., Colbourn C.J., Dukes P.: Constructions for permutation codes in powerline communications. Des., Codes Cryptogr. **32**, 51–64 (2004).
8. Colbourn C.J., Kløve T., Ling A.C.H.: Permutation arrays for powerline communication and mutually orthogonal latin squares. IEEE Trans. Inform. Theory **50**, 1289–1291 (2004).
9. Deza M., Vanstone S.A.: Bounds for permutation arrays. J. Statist. Plann. Inference **2**, 197–209 (1978).
10. Dukes P., Sawchuck N.: Bounds on permutation codes of distance four. J. Algebr. Comb. **31**, 143–158 (2010).
11. Frankl P., Deza M.: On maximal numbers of permutations with given maximal or minimal distance. J. Combin. Theory Ser. A **22**, 352–360 (1977).
12. Han Vinck A.J.: Coded modulation for power line communications. A.E.Ü. Int. J. Electron. Commun. **54**(1), 45–49 (2000).
13. Huczynska S.: Powerline communications and the 36 officers problem. Phil. Trans. R. Soc. A. **364**, 3199–3214 (2006).
14. Hulpke A.: Constructing transitive permutation groups. J. Symbolic Comput. **39**(1), 1–30 (2005).
15. Hurley S., Smith D.H., Thiel S.U.: FASoft: A system for discrete channel frequency assignment. Radio Sci. **32**(5), 1921–1939 (1997).
16. Janiszczak I., Staszewski R.: An improved bound for permutation arrays of length 10. http://www.iem.uni-due.de/preprints/IJRS.pdf (downloaded 1st March 2011).
17. Konc J., Janežič D.: An improved branch and bound algorithm for the maximum clique problem. MATCH communications in mathematical and in computer chemistry **58**, 569–590 (2007).
18. Montemanni R., Smith D.H.: Heuristic algorithms for constructing binary constant weight codes. IEEE Trans. Inform. Theory **55**(10), 4651–4656 (2009).
19. Östergård P.R.J.: A new algorithm for the maximum-weight clique problem. Nordic J. Comput. **8**(4), 424–436 (2001).
20. Östergård P.R.J.: A fast algorithm for the maximum clique problem. Dis. Appl. Math. **120**, 197–207 (2002).
21. Pavlidou N., Han Vinck A.J., Yazdani J., Honary B.: Power line communications: state of the art and future trends. IEEE Commun. Mag. **41**(4), 34–40 (2003).
22. Tarnanen H.: Upper bounds on permutation codes via linear programming. Eur. J. Combin. **20**, 101–114 (1999).