CONTEMPORARY COMPUTER-AIDED CONSTRUCTION, COUNTING, CLASSIFICATION, AND CHARACTERIZATION OF COMBINATORIAL CONFIGURATIONS

Patric R. J. Östergård
Department of Communications and Networking
Aalto University
P.O. Box 13000, 00076 Aalto, Finland
E-mail: `patric.ostergard@tkk.fi`

Joint work with Alexander Hulpke, Petteri Kaski, Veli Mäkinen, Kevin Phelps, and Olli Pottonen.

The research was supported in part by the Academy of Finland.

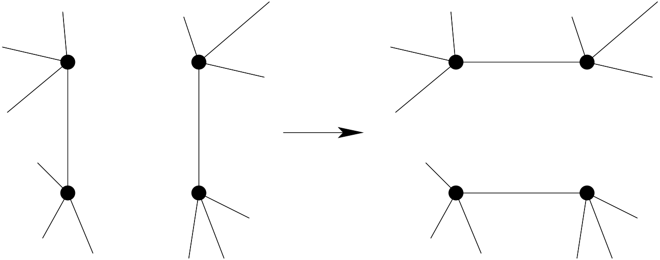
Outline of Talk

1. Construction and Characterization, via *switching*.
2. Classification, via *exhaustive search*.
3. Counting, via *dynamic programming*.

Switching

Switching is a local transformation that leaves the main (basic as well as regularity) parameters of a combinatorial object unchanged.

Example. 2-switch of a graph.



History of Switching

Norton (1939) and Fisher (1940) Latin squares and Steiner triple systems [F,N].

Vasil'ev (1962) (Perfect) codes [V].

Van Lint and Seidel (1966) : Graphs (*Seidel switching*) [LS].

[F] R. A. Fisher, An examination of the different possible solutions of a problem in incomplete blocks, *Ann. Eugenics* **10** (1940), 52–75.
[N] H. W. Norton, The 7 x 7 squares, *Ann. Eugenics* **9** (1939), 269–307.
[V] Ju. L. Vasil'ev, On nongroup close-packed codes, (in Russian), *Problemy Kibernet.* **8** (1962), 337–339.
[LS] J. H. van Lint and J. J. Seidel, Equilateral point sets in elliptic geometry, *Indag. Math.* **28** (1966), 335–348.

Why Switch?

There are many reasons for switching, including the following:

1. As a part of a mathematical proof.
2. To define neighbors in a local search algorithm.
3. To try to find new combinatorial objects from old ones.
4. In order to gain understanding in why there are so many equivalence/isomorphism classes of objects with certain parameters.

Switching Unrestricted Codes

All codes in the sequel are *binary*.

Example. Code with minimum distance 3.

$$\begin{array}{c}
000000001111111 \\
011101001000110 1 \\
000111100011100 1 \\
001100111001001 1 \\
001011010101010 1 \\
011010100100101 1 \\
010110010010011 1
\end{array}$$

Switching Unrestricted Codes

All codes in the sequel are *binary*.

Example. Code with minimum distance 3.

```
000000011111111
011101001000110 1
```

Switching Unrestricted Codes

All codes in the sequel are *binary*.

Example. Code with minimum distance 3.

```
0000010011111111
0111010010001101
0001111000111001
0011001110010011
0010110101010101
0110101001001011
0101100100100111
```

Switching Unrestricted Codes

All codes in the sequel are *binary*.

Example. Code with minimum distance 3.

000001001111111
011101001000110
000111100011100
001100111001001
001011010101010
011010100100101
010110010010011

Switching Unrestricted Codes

All codes in the sequel are *binary*.

Example. Code with minimum distance 3.

000001001110011
011101001000110 1

Switching Unrestricted Codes

All codes in the sequel are *binary*.

Example. Code with minimum distance 3.

0000010011100011
0111010010001101
0001111000111001
0011001110010011
0010110101010101
0110101001001011
0101100100100111

Switching Unrestricted Codes

All codes in the sequel are *binary*.

Example. Code with minimum distance 3.

010001111110011
011101001000110 1
000111100011100 1
001100111001001 1
001011010101010 1
011010100100101 1
010110010010011 1

Switching Unrestricted Codes

All codes in the sequel are *binary*.

Example. Code with minimum distance 3.

```
0100011111100011
0111010010001101
0001111000111001
0011001110010011
0010110101010101
0110101001001011
0101100100100111
```

Switching Unrestricted Codes

All codes in the sequel are *binary*.

Example. Code with minimum distance 3.

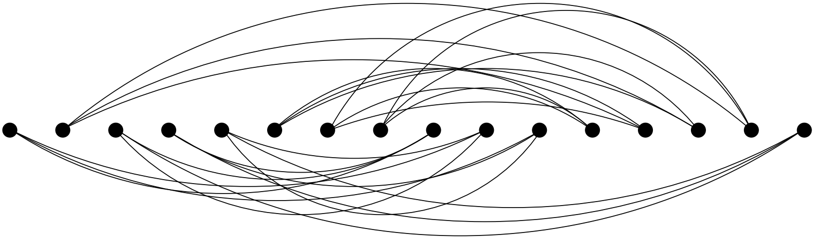
```
0100011111100001
0111010010001101
0001111000111001
0011001110010011
0010110101010101
0110101001001011
0101100100100111
```

Switching via an Auxiliary Graph

1. Consider a particular coordinate i.
2. Construct a graph G with one vertex for each codeword and an edge between two vertices that differ in the ith coordinate and whose mutual distance equals the minimum distance of the code.
3. Complement the ith coordinate in a connected component of the graph G.

Example: Auxiliary Graph

For the previous example we get the following auxiliary graph with respect to the first coordinate:



Switching Graph and Switching Classes

Switching graph: A graph with one vertex for each equivalence class of codes and with an edge if there is a switch taking a code from one class to the other.

Switching class: A connected component of the switching graph, in other words, a complete set of (equivalence classes of) codes connected via a sequence of switches.

Example: Switching Error-Correcting Codes

n	d	$A(n,d)$	N	Sizes of switching classes
6	3	8	1	1
7	3	16	1	1
8	3	20	5	3, 2
9	3	40	1	1
10	3	72	562	165, 134, 110, 89, 26, 15, 14, 9
11	3	144	7398	7013, 385

Switching Constant Weight Codes

The aforementioned switch changes the weight of codewords.

$\Rightarrow$

If we consider codes with constant Hamming weight, then we need to apply a switch in a different way.

How?

Modification to Basic Switch

0000111
1100100
0110001
0011100
0101010
1010010
1001001

Modification to Basic Switch

```
0000111
1100100
0110001
0011100
0101010
1010010
1001001
```

Modification to Basic Switch

1000111
0100100
0110001
0011100
0101010
1010010
1001001

Modification to Basic Switch

1000111
0100100
0110001
0011100
0101010
1010010
1001001

Modification to Basic Switch

1000100
0100111
0110001
0011100
0101010
1010010
1001001

Modification to Basic Switch

1000100
0100111
0110001
0011100
0101010
1010010
1001001

Modification to Basic Switch

1100100
0000111
0110001
0011100
0101010
1010010
1001001

Switching Designs

What we just saw is the well-known **Pasch switch** for designs!
General approach for *Steiner systems*:

1. Consider two points i and j.
2. Construct a graph G with a vertex for each block that contains exactly one of the points i, j and with edges between blocks whose intersection contains neither i nor j and that are "at minimum distance".
3. Permute the points i and j in a connected component of G (actually: complement).

A switch is a particular **trade**!

Example: Switching Steiner Systems

The switching graph of the 11 084 874 829 isomorphism classes of Steiner triple systems of order 19 is connected.

In fact, even the switching graph of the labeled 1 348 410 350 618 155 344 199 680 000 designs is connected.

This work was computationally rather challenging (required a lot of memory).

The 1 054 163 isomorphism classes of Steiner quadruple systems of order 16 belong to switching classes of size 1 043 486, 1 853, 951, 920, 676, 584, 495, 427,. . . ,1.

[KMO] P. Kaski, V. Mäkinen, and P. R. J. Ö., The cycle switching graph of the Steiner triple systems order 19 is connected, submitted for publication.

Switching Covering Codes

A covering code has the property that all words in the ambient space are within Hamming distance R from some codeword.

How to switch a covering code with codewords $\mathbf{c} = (c_1, c_2, \ldots, c_n)$ in some coordinate s ?

Criterion for edges in auxiliary graph:

$$d_H(\mathbf{c}, \mathbf{c}') \leq 2R + 1, \quad d_H(\mathbf{c}, \mathbf{c}') \text{ odd}, \quad c_s \neq c'_s, \tag{1}$$

Switching Covering Codes: Outline of Proof

It suffices to consider a word **b** that is at distance R from a codeword **a** that is altered and the case when $a_s = b_s$.

Consider the word **c**, which coincides with **b**, except that $a_s = b_s \neq c_s$; and also consider the word **e** that covers **c**. We get three cases:

1) $e_s = b_s$: $\Rightarrow d_H(\mathbf{b}, \mathbf{e}) \leq R - 1$.
2) $e_s \neq b_s$ and $d_H(\mathbf{c}, \mathbf{e}) \leq R - 1$: . . .
3) $e_s \neq b_s$ and $d_H(\mathbf{c}, \mathbf{e}) = R$: $\Rightarrow d_H(\mathbf{a}, \mathbf{e})$ is odd and smaller than or equal to $R + 1 + R = 2R + 1 \Rightarrow$ the conditions of (1) are fulfilled.

Example: Switching Covering Codes

n	R	$K(n,R)$	N	Sizes of switching classes
5	1	7	1	1
6	1	12	2	2
7	1	16	1	1
8	1	32	10	5, 3, 2

The two known codes attaining $K(9,1) = 62$ belong to one switching class.

1-Perfect Codes

The **1-perfect codes** are error-correcting codes with minimum distance 3 and covering codes with covering radius 1. They exist for all lengths $n = 2^i - 1$. We consider $n = 15$; then there are $2^{11} = 2048$ codewords.

Fact. The codewords at distance 3 from a codeword of a 1-perfect code form a *Steiner triple system*. The codewords at distance 4 from a codeword of an extended 1-perfect code form a *Steiner quadruple system*.

Classifying the 1-Perfect Codes of Length 15

1. Consider the objects obtained by puncturing the 1 054 163 Steiner quadruple systems of order 16 [KOP].
2. For any such *seed* and the all-zero word (141 words in total), exhaustively search for the remaining $2\,048 - 141 = 1\,907$ codewords (instances of *exact cover*).
3. Extend the solutions to length 16.
4. Carry out isomorph rejection.

[KOP] P. Kaski, P. R. J. Ö., and O. Pottonen, The Steiner quadruple systems of order 16, *J. Combin. Theory Ser. A* 113 (2006), 1764–1770.

Classification Result

There are 5 983 inequivalent binary 1-perfect codes of length 15; these have 2 165 inequivalent extensions [OP].

The sizes of the switching classes have been determined in [OPP]: 5 819, 153, 3, 2, 2, 1, 1, 1, and 1.

[OP] P. R. J. Ö. and O. Pottonen,The perfect binary one-error-correcting codes of length 15: Part I—Classification, *IEEE Trans. Inform. Theory* **55** (2009), 4657–4660, 2009. Codes at arXiv:0806.2513v3.

[OPP] P. R. J. Ö., O. Pottonen, and K. T. Phelps, The perfect binary one-error-correcting codes of length 15: Part II—Properties, *IEEE Trans. Inform. Theory*, to appear.

Shortened Perfect Codes

Theorem A. (Best & Brouwer, 1977) When 1-perfect codes are shortened once, twice, or three times, one gets optimal one-error-correcting codes.

Theorem B. (Blackmore, 1999) The inverse of Theorem A holds for codes with the parameters of 1-perfect codes shortened once.

Theorem C. (Ö. & Pottonen [OP]) The inverse of Theorem A does not always hold for codes with the parameters of 1-perfect codes shortened twice.

Proof. Switching the codes obtained by shortening the 1-perfect codes of length 15 twice gives two new codes.

[OP] P. R. J. Ö. and O. Pottonen, Two optimal one-error-correcting codes of length 13 that are not doubly shortened perfect codes, *Des. Codes Cryptogr.*, to appear.

Counting with the Orbit-Stabilizer Theorem

The **Orbit-Stabilizer Theorem** can sometimes be used to count the number of isomorphism classes faster than they can be generated:

$$|\Omega| = |\Gamma| \sum_i \frac{N_i}{i}$$

Γ	a finite group
Ω	a finite set on which Γ acts
N_i	the number of orbits on Ω whose elements have stabilizer subgroups of order i in Γ.

From $|\Omega|$ and N_2, N_3,..., we can easily calculate N_1 and thereby obtain $N = \sum_i N_i$.

Counting Latin Squares

The following technique for counting the number of Latin squares of side n, that is, $|\Omega|$, is well known [MW].

- Approach the problem via 1-factorizations of $K_{n,n}$.
- A set of k 1-factorizations of $K_{n,n}$ can be obtained as a union of $k-1$ 1-factorizations with one more 1-factorization $\Rightarrow$ dynamic programming possible.
- A set of k 1-factorizations of $K_{n,n}$ form a k-regular bipartite graphs. It suffices to maintain counts for (isomorphism class representatives of) such regular graphs.

[MW] B. D. McKay and I. M. Wanless, On the number of Latin squares, *Ann. Comb.* **9** (2005), 335–344.

Counting Latin squares of side 11

For $n = 11$, we know $|\Omega|$, N_3, $N_4, \dots$, that is, everything but N_1 and N_2.

Idea. Count N_2 in a way that is analogous the aforemention technique for obtaining $|\Omega|$.

This idea, which is straightforward on a principal level, but involves MANY details and subcases, was implemented in [HKO].

[HKO] A. Hulpke, P. Kaski, and P. R. J. Ö, The number of Latin squares of order 11, *Math. Comp.*, to appear.

The Counts

There are

- 2 036 029 552 582 883 134 196 099 main classes of Latin squares of order 11;
- 6 108 088 657 705 958 932 053 657 isomorphism classes of one-factorizations of $K_{11,11}$;
- 12 216 177 315 369 229 261 482 540 isotopy classes of Latin squares of order 11;
- 1 478 157 455 158 044 452 849 321 016 isomorphism classes of loops of order 11; and
- 19 464 657 391 668 924 966 791 023 043 937 578 299 025 isomorphism classes of quasigroups of order 11.

A Final Comment

The main contribution here is not the *numbers* but the *algorithms*.

Example. (Work in progress) The number of Hamilton cycles of a graph can be counted via dynamic programming. If this count is greater than 0, then the graph is Hamiltonian. NP-complete problem!