Computing binary combinatorial gray codes via exhaustive search with SAT solvers

Author(s):
Zinovik, Igor; Kröning, Daniel; Chebiryak, Yury

Publication Date:
2008-04

Permanent Link:
https://doi.org/10.3929/ethz-a-005706906

Originally published in:

Rights / License:
In Copyright - Non-Commercial Use Permitted
The reduction from Theorem 1.1 to Lemma 1.1 utilizes Jensen’s inequality, majorization (Marshall and Olkin, [2]), and a conditioning argument; details can be found in Ordentlich [1].

Lemma 1.1 was proven by Ordentlich [1] using an involved combinatorial argument. It is the purpose of this correspondence to deliver an elementary proof of Lemma 1.1, thereby giving an alternative short derivation of Theorem 1.1.

II. A SHORT PROOF OF LEMMA 1.1

We shall perform induction \( n \). It is simple to verify that (1) is true for \( n = 2 \). Suppose it holds for \( n < m \). When \( n = m \geq 3 \), without loss of generality assume \( a_1 \leq a_2 \leq \cdots \leq a_m \), and consider the nontrivial case of \( 0 < a_m \leq 1 \). Let \( j \) be an integer such that \( k = m - 2j > 0 \). Denote \( S = \sum_{i=1}^{m-1} Z_i a_i / a_m \). We have

\[
\Pr(\sum_{i=1}^{m} Z_i a_i \in [-k, k]) \\
= \Pr(\sum_{i=1}^{m} Z_i + S \in [-k/a_m, k/a_m]) \\
\geq \Pr(\sum_{i=1}^{m} Z_i + S \in [-k, k]) \\
= \frac{1}{2} \left[ \Pr(S \in [-k-1, k-1]) + \Pr(S \in [-k+1, k+1]) \right] \\
= \frac{1}{2} \left[ \Pr(S \in [-k-1, k+1]) + \Pr(S \in [-k+1, k-1]) \right] \\
where the formula \( \Pr(A) + \Pr(B) = \Pr(A \cup B) + \Pr(A \cap B) \) is used in the last equality.

Denote \( S' := \sum_{i=1}^{m-2} Z_i \). If \( k - 1 > 0 \), then by the induction hypothesis (recall that \( k - 1 = m - 1 - 2j \))

\[
\Pr(S \in [-k+1, k-1]) \geq \Pr(S' \in [-k+1, k-1]). \tag{2}
\]

If \( k - 1 = 0 \) then \( m \) is an odd integer. Notice that (2) is still valid because the right-hand side is zero. Similarly

\[
\Pr(S \in [-k-1, k+1]) \geq \Pr(S' \in [-k-1, k+1]).
\]

Together we have, as long as \( k = m - 2j > 0 \)

\[
\Pr(\sum_{i=1}^{m} Z_i a_i \in [-k, k]) \\
\geq \frac{1}{2} \left[ \Pr(S' \in [-k-1, k-1]) + \Pr(S' \in [-k+1, k+1]) \right] \\
= \frac{1}{2} \left[ \Pr(S' \in [-k-1, k+1]) + \Pr(S' \in [-k+1, k-1]) \right] \\
= \Pr(S_{m-1} + \sum_{i=1}^{m-2} Z_i \in [-k, k])
\]

where for the last equality we consider the two cases \( S_{m-1} = 1 \) and \( S_{m-1} = -1 \). Thus, the claim holds when \( n = m \), and Lemma 1.1 is proven.

ACKNOWLEDGMENT

The author thanks the Associate Editor and a referee for their valuable comments.

REFERENCES

The objective of this correspondence is the evaluation of SAT-based techniques for the construction of binary combinatorial codes. The correspondence is organized as follows: In Section II, we present an encoding of a binary code called the coil-in-the-box code [7] as a Boolean formula and the application of SAT solvers to classify such codes with respect to symmetry transformations of hypercubes. This classification is of interest for the Glass model of gene regulatory networks [8]. In Section III, we extend the proposed encoding to the construction of a generalization of coil-in-the-box codes called circuit codes [9], and distance-preserving codes [10]. In Section IV, we present a SAT-based method for generating a subset of the circuit codes that are used in analog-to-digital conversion devices [2], and a Gray code for necklaces longer than those presented in [1]. The tables in the Appendix summarize the results of our experiments and report the coordinate sequences of the new codes.

II. COIL-IN-THE-BOX CODES

A. SAT Encoding

A coil in a graph is a simple cycle without chords, which is defined as follows.

Definition 1 (Coil-in-the-Box [7]): A simple cycle in an n-dimensional cube is called a coil-in-the-box if every edge in the n-cube that joins two vertices of the cycle is an edge of this cycle.

If the nodes of an n-dimensional unit cube are labeled by their coordinates, a coil-in-the-box is represented by a Gray code of binary words \( W_i \) of length \( n \) with \( i = 1, \ldots, N \). The number \( N \) of words in the code is called the period of the code. The codes of the maximum period with \( n \) up to 6 were generated by a computer search [7, 11]. The construction of the codes of maximum period with words of length 7 shown in [12] requires a restriction of the search to the codes that contain a sequence of \( n + 1 \) words \( W_1, W_2, \ldots, W_{n+1} \) such that the Hamming distance \( d_{ij}(W_i, W_{i+n}) \) equals \( n \) (we will call such a sequence a diagonal of the n-cube). The exact value of the maximum period is unknown for \( n \geq 8 \).

The lower and upper bounds for the maximum period is a function of \( n \) [7]. The most recent results provide only lower bounds for the maximal period and generate the codes via evolutionary techniques [13].

In order to apply SAT solvers to generate codes, we define Boolean variables \( X_{ij} \) and let an position \( j \) of the binary word number \( i \). Auxiliary Boolean variables \( H^1_{ij} \) and \( H^2_{ij} \) encode the Hamming distance \( d_{ij}(W_i, W_j) \) between words \( W_i \) and \( W_j \) such that the true value of \( H^2_{ij} \) with \( \alpha \in \{0, 1\} \) determines that the words are identical or the distance equals 1, respectively. The following example illustrates the construction of a SAT instance for the search of a Gray code with period 3 which is composed of word of length 2.

Example: Construction of SAT instance for the search for a Gray code with \( n = 2 \) and \( N = 3 \).

The auxiliary variables are defined as follows:

\[
H^1_{13} := (X_{11} \leftrightarrow X_{31}) \land (X_{12} \leftrightarrow X_{32})
\]

\[
H^1_{12} := ((X_{11} \leftrightarrow \neg X_{21}) \land (X_{12} \leftrightarrow X_{22}) \lor (X_{11} \leftrightarrow X_{21}) \land (X_{12} \leftrightarrow \neg X_{22}))
\]

\[
H^1_{13} := ((X_{11} \leftrightarrow \neg X_{31}) \land (X_{12} \leftrightarrow X_{32}) \lor (X_{11} \leftrightarrow X_{31}) \land (X_{12} \leftrightarrow \neg X_{32}))
\]

\[
H^1_{23} := ((X_{21} \leftrightarrow X_{31}) \land (X_{22} \leftrightarrow X_{32}) \lor (X_{21} \leftrightarrow X_{31}) \land (X_{22} \leftrightarrow \neg X_{32}))
\]

\[
H^1_{12} := (X_{11} \leftrightarrow \neg X_{21}) \land (X_{12} \leftrightarrow X_{22}) \lor (X_{11} \leftrightarrow X_{21}) \land (X_{12} \leftrightarrow \neg X_{22})
\]

Thus, the condition that the distance between the words equals one and the first and the last words are distinct, is written as follows:

\[
H^1_{12} \land H^1_{13} \land H^1_{23} \land \neg H^1_{13}.
\]

This formula can be used as input to a SAT solver. SAT solvers determine whether the propositional formula is satisfiable or not and in the former case, return a satisfying assignment to the variables. The output obtained is decoded into the sequence of the binary words.

The construction above yields \( N \times n \) independent variables \( X_{ij} \), which correspond to the nodes of the cycle. We also tested an alternative encoding using the coordinate sequences instead of the node variables \( X_{ij} \). A coordinate sequence is a list \([s_i, s_j, \ldots, s_{n}]\), \( i \in \{1, \ldots, N - 1\} \) of the unique coordinates in which \( W_i \) and \( W_{i+1} \) differ. Thus, the code is uniquely defined by a choice of the first word \( W_i \) and a coordinate sequence \([s_i, s_j, \ldots, s_n]\).

For encoding coordinate sequences, every binary integer \( s_i \) is represented by \([\log(n)] \) Boolean variables \( S_{i,0} \) with \( \log(n) \). The true valuation of \( S_{i,0} \) denotes a 1 in position \( q \) of the binary word \( s_i \), while the false encodes a 0. In this encoding, the number of the variables is \( (N - 1) \cdot \log(n) + n \) variables, where \( X_{ij} \) is defined recursively as a function of \( S_{i+1,0} \) and \( X_{j+1,0} \)

\[
X_{ij} := \begin{cases} \neg X_{i+1,j} & \text{if } s_{i+1} = j \\ X_{i+1,j} & \text{otherwise.} \end{cases}
\]

The computation of the codes was carried out using a PC with an Intel Xeon (3.0-GHz, 4-GB RAM, running Linux) with a timeout of 24 h. The satisfying assignments were obtained using the SMT-solver Yices [15] for codes with codeword length up to 8. While the coils computed did not surpass the longest known coil of period 96, the experiments indicate that the codes with periods up to 82 can be generated within 6 h. The results show that the sequence encoding reduces the run time for the cases without satisfying assignments (the UNSAT cases), e.g., in case \( n = 5 \), \( N = 16 \), UNSAT for the node encoding is returned in 257 s while the sequence encoding needs only 0.4 s. The node encoding was faster for all tests with satisfying assignments except for the maximum coil in dimension 7 (the runtime is 36 min compared to 162 min for the node encoding).

Unlike the search method for the maximum known codes described in [12], the presented construction allows for the generation of maximum coils without the cube diagonals (a longest coil in dimension 7 that has no cube diagonal is shown in the Appendix). The suggested method relies on a direct encoding of the code definition without any additional restrictions, and thus, the search space contains all coils for a prescribed dimension and length. The UNSAT cases obtained as a result of incrementally increasing the period \( N \) serve as a proof that the previous satisfying assignment constitutes a maximum-period code for the prescribed dimension.

B. Classification of Codes in Dimension 6 and 7

The absence of restrictions narrowing the solution space makes it possible to use the encoding to analyze all combinatorial codes for a prescribed dimension. The classification of the coils-in-the-box codes with respect to axis permutations of the n-cube is of interest in Glass models for neural and gene regulatory networks. In this model, active and inactive states of a particular gene are depicted by 1/0, and a binary codeword represents a set of the genes in a cell at a given time instant. Coil-in-the-box codes correspond to stable periodic processes which describe biologically relevant dynamics of the gene sets. Thus, the number of the equivalence classes of the codes indicates how many different types of cells can be regulated by the set of the genes.

The number of the equivalence classes up to dimension 5 is computed in [8]. The lower bounds for the number of the classes in dimension 6 has been obtained in [16]. The presented classification algorithm is a modified ALL-SAT procedure as described in [16]. Every assignment obtained is a representative of an equivalence class. MiniSAT was used for the computations, and the results are summarized in Table I.
TABLE I
THE NUMBER OF EQUIVALENCE CLASSES FOR THE COILS-IN-THE-BOX FOR DIMENSION 6

<table>
<thead>
<tr>
<th>Length</th>
<th>Lower bound of [16]</th>
<th>#Equivalence-classes</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>385</td>
<td>563</td>
</tr>
<tr>
<td>18</td>
<td>1066</td>
<td>1228</td>
</tr>
<tr>
<td>20</td>
<td>981</td>
<td>1032</td>
</tr>
<tr>
<td>22</td>
<td>465</td>
<td>478</td>
</tr>
<tr>
<td>24</td>
<td>103</td>
<td>110</td>
</tr>
<tr>
<td>26</td>
<td>4</td>
<td>4</td>
</tr>
</tbody>
</table>

The coordinate sequences for the equivalence classes of the longest coils in dimension 6 are presented in the Appendix. One of four classes is found to contain no diagonal, and one class represents the coils with periodic coordinate sequences.

The exhaustive search for all maximum coils in dimension 7 is known to be computationally demanding [12]: the search for coils with a cube diagonal took more than one month on a network consisting of five SUN Microsystems SparCenter 1000’s with two processors each. We have applied the propositional encoding described above for the classification of the maximum coils ($N = 48$) in dimension 7. The full classification could not be archived within a timeout of 100 days. The number of the equivalence classes found within the timeout is 126, where 74 classes do not have the cube diagonal, i.e., they cannot be found by the algorithm used in [12].

III. NEW CIRCUIT CODES, AND DISTANCE-PRESERVING CODES

The SAT encoding described above can be modified for the construction of circuit codes [9] with different spreads and distance-preservation codes [10].

A. Circuit Codes

Circuit codes are defined as a generalization of the coil-in-the-box definition (1) as

$$\forall k, l : d_H(W_k, W_l) < \delta \Rightarrow d_C(W_k, W_l) < \delta$$ \hspace{1cm} (2)

where the positive integer $\delta$ is called the spread of the code [17]. Thus, the circuit codes with the spread $\delta = 2$ are the coils-in-the-box, and the codes with $\delta = 1$ of period $N = 2^n$ are the Hamiltonian cycles in the $n$-cube. Moreover, any code with spread $\delta_1$ is a circuit code with spread $\delta_2$ if $\delta_1 \geq \delta_2$ [17]. This property implies that the period of maximum circuit codes with a spread $\delta_1$ does not exceed the maximum period of the codes with $\delta_2 \leq \delta_1$.

The construction of the longest known circuit codes is based on either an exhaustive search or an algorithm that restricts the search to the codes with periodic coordinate sequences [9]. The codes can be generated from the existing ones from lower dimensions $n$ (e.g., [18]).

We conducted computations aiming to improve the results in [9], which present the longest known circuit codes with spread $\delta \leq 7$. MiniSAT finds a satisfying assignment for $n = 11$, $\delta = 4$, and $N = 60$ within 23 min. Six codes with greater periods were obtained within the timeout of 24 h. The coordinate sequences of the found codes are presented in the Appendix (see Table IV).

To the best of our knowledge, there are no circuit codes reported for the spreads $\delta > 7$. The SAT encoding was used to compute codes with spread $\delta$ up to 17. We present four codes that are longer than known lower bounds of [17] in the Appendix. We also show four codes with the periods whose optimality was proved in [17].

B. Distance-Preserving Codes

The distance-preserving codes are defined by the following equation:

$$\forall k, l : d_C(W_k, W_l) \leq \delta \Rightarrow d_H(W_k, W_l) = d_C(W_k, W_l)$$ \hspace{1cm} (3)

This code preserves the Hamming distance between the codewords for all distances up to a threshold $m$. The distance-preserving code in dimension $n$ is denoted as $(m, n)$-code, and the $(m, n)$-codes with $N = 2^n$ are called the complete codes. Two types of algorithms are reported for distance-preserving codes. The method shown in [10] generates the codes only of a certain length $L = m2^{n-\lceil \log_2 m \rceil}$. The algorithm proposed in [19] constructs $(n-1, n)$-codes with period $(n-1)2^{\lfloor n/2 \rfloor}$.

In contrast to the known methods, the SAT encoding is easy to modify for the construction of the codes with an arbitrary prescribed codeword length $n$ and threshold $m$. As examples, we computed the codes $(6, 7)$ with a maximum period of 100, and $(7, 8)$ with a period of 126 (see the Appendix). While the values of the maximum periods and the lower bounds for the periods of such codes were calculated in [19], to the best of the authors’ knowledge, the coordinate sequences for these codes were not yet reported in the literature.2

IV. NEW SINGLE-TRACK CIRCUIT CODES AND NECKLACES

Single-track circuit codes introduced in [21] are a subclass of circuit codes which are used for digital–to–analog conversion. The single-track codes are circuit codes that possess an additional property defined in terms of component sequences [2].

Definition 2 (Component Sequence): Let $C$ be a cyclic path on the $n$-cube consisting of $N$ binary codewords $W_1, \ldots, W_N$, where $W_i \in \{w_1, \ldots, w_N\}$. The component sequence $j$ of $C$, denoted $C/j$, is the binary periodic sequence $w_{1j}, \ldots, w_{nj}$ consisting of component $j$ of each of the codewords of $C$ ($1 \leq j \leq n$).

The formal definition of the single-track codes reads as follows. 

Definition 3 (Single-Track Circuit Codes): Let $C$ be a circuit code with period $N$, spread $\delta$, composed of codewords of length $n$. Then $C$ is said to be a single-track circuit code if its component sequence $C/j$ is a cyclic shift of sequence $C/1$ for each $2 \leq j \leq n$. We denote these codes by $(n, \delta, N)$.

For every $n$, there is a $(n, n, 2n)$ code called the trivial code that has the coordinate sequence composed of two repeating $n$-cube diagonals. Single-track circular codes of longer periods are constructed in [2] by embedding a set of the codewords into the known circuit codes. The codes are generated up to $\delta \leq 6$ using the circuit codes from [9] that have the spread $\delta$ bounded by 7. A single-track circuit code is called optimal if there is no single-track circuit code with the same period but composed of codewords of shorter length. Two conditions were used in [2] to determine the optimality of the codes: a) the code period does not exceed the period of the corresponding circuit codes, and b) if a single-track code exists, its period equals an even multiple of the codeword length.

2Three months after the manuscript was submitted to IEEE TRANSACTIONS OF INFORMATION THEORY, code $(7, 8)$ with a period of 210 was reported in [20].
We conducted computations aimed to construct single-track circuit codes with the periods longer than the results in [2]. No satisfying assignments were obtained within a 24-h timeout. UNSAT was returned for five test cases: /104/55/59/51/59/50/56/105, /104/56/59/51/59/50/52/105, and /104/57/59/51/59/55/50/105, which is greater than in [2], were computed for /110/61/49/48/59/49/49, and /78/61/50/110 within 2 h. The corresponding coordinate sequences (see the Appendix) indicate that the obtained codes are nontrivial. We also conducted a search for nontrivial single-track circuit codes for /110/61/55/59/56/59 and /49/50. The results of the computations show that there are no nontrivial single-track circuit codes in these cases.

The longest known circuit and single-track circuit codes have been constructed by restricting the search to the codes that possess various internal symmetries. The methods rely on the generation of the Gray-ordered binary necklaces as a first step of the construction [9]. An /110/-bead binary necklace is an equivalence class of binary /110/-tuples under rotation [1]. Necklace-based construction of the codes is proved in [22], [23] to be very successive for small /110/. The methods are not easily adapted to produce the codes with /110/ larger than /55/ because of a rapid increase of the number of necklaces.

While several known algorithms provide a complete list of the necklaces with a prescribed codeword length /n/, none of them computes a Gray code for necklaces. Efficient algorithms producing Gray-ordered necklaces are of interest in combinatorics [24], and the question whether a complete list of Gray-ordered necklaces exists for /n/ > /7/ is among the open problems of combinatorial Gray codes (a parity argument shows that this is impossible for an even /n/) [1].

The results of our computations for six-bead necklaces indicate that there is no Gray code with length greater than 13 codewords. The Gray codes for eight-bead necklaces were obtained up to 33 codewords. The complete list of nine-bead necklaces contains 60 codewords [25], thus, a propositional formula describing the list was generated and consequently used as input to the SAT solver. A satisfying assignment has been obtained within 68 min using the encoding of the coordinate sequences. The coordinate sequence of the Gray code is presented in the Appendix. To the best of our knowledge, the Gray code consisting of all nine-bead necklaces was not yet reported in the literature.

V. CONCLUSION

We present a propositional encoding for the generation of coil-in-the-box codes, circuit codes, distance-preserving codes, single-track codes, and necklaces. The method we suggested for the construction of the codes utilizes efficient backtracking algorithms implemented in state-of-the-art propositional SAT solvers. The encoding enforces an exhaustive search over all codes satisfying the definition, and thus it can be used for a classification of the codes. Search within a desired subclass of the codes can be conducted by using additional blocking clauses.

We report new lower bounds for ten circuit codes. We present three new nontrivial single-track circuit codes and two new distance-preserving codes. An advantage of the SAT-based approach is that a SAT solver returns a definite answer whether a code with given parameters exists or not. The negative answer may serve as a proof of optimality of the codes.
Using this approach, we proved optimality for five known single-track circuit codes.

We found all equivalence classes with respect to the $n$-cube symmetry transformations for the coil-in-the-box codes in dimension 6, and obtained a lower bound, namely 126, on the number of equivalence classes for longest coils in dimension 7. We also proved by construction the existence of the Gray code for the complete list of nine-bead necklaces, thus improving a known result for seven-bead necklaces.

APPENDIX

The results of our experiments and the coordinate sequences of the new codes are shown in Tables II–VII.

ACKNOWLEDGMENT

The authors are indebted to an anonymous reviewer for bringing the papers [21]–[23] to their attention and for many helpful comments and suggestions that improved this paper. The authors also appreciate detailed answers of A. L. Perezhogin regarding the distance-preserving codes.

REFERENCES